Verity Whitepaper  
Beta 2.0

A DECENTRALIZED GOVERNANCE AND REPUTATION SYSTEM FOR BUILDING VERIFIABLE, TRUSTWORTHY AND COLLABORATIVE COMMUNITIES

Matt Goldenberg, Michael Hoard, Doug King, Eric Johnsohn, Keith Swallow

Table of Contents

[Note 5](#_Toc467091473)

[Special Thanks 5](#_Toc467091474)

[Abstract 6](#_Toc467091475)

[Background 6](#_Toc467091476)

[Crowdsourcing 6](#_Toc467091477)

[Wisdom of the Crowds 6](#_Toc467091478)

[Internet Reputation 7](#_Toc467091479)

[Combining Reputation and Crowdsourcing 7](#_Toc467091480)

[Ethereum 7](#_Toc467091481)

[Summary of the Verity Platform 7](#_Toc467091482)

[Reputation Layer 8](#_Toc467091483)

[Governance Layer 9](#_Toc467091484)

[Collaboration Layer 9](#_Toc467091485)

[Application Layer 9](#_Toc467091486)

[An Example 10](#_Toc467091487)

[Reputation Layer 10](#_Toc467091488)

[Reputation and Ratings Standard 11](#_Toc467091489)

[Reputation Traits 11](#_Toc467091490)

[Ratings and Attestations 12](#_Toc467091491)

[Reputation INstances 12](#_Toc467091492)

[Reputation Classes 12](#_Toc467091493)

[Values-Based Reputation 12](#_Toc467091494)

[Value Tags 13](#_Toc467091495)

[Value Scores 14](#_Toc467091496)

[Skills-Based Reputation 17](#_Toc467091497)

[Skill Tags 18](#_Toc467091498)

[Skill Tokens 18](#_Toc467091499)

[Verity Campaigns and Contests 22](#_Toc467091500)

[Contribution-Based Reputation 30](#_Toc467091501)

[Utility Function Standard 31](#_Toc467091502)

[Contribution Tokens 31](#_Toc467091503)

[Governance Layer 32](#_Toc467091504)

[Governance Standard 33](#_Toc467091505)

[Actions 33](#_Toc467091506)

[Proposals 33](#_Toc467091507)

[Roles 34](#_Toc467091508)

[Permissions 34](#_Toc467091509)

[Governance Systems 34](#_Toc467091510)

[Value-Based Hierarchy 34](#_Toc467091511)

[Contribution-Based Voting 35](#_Toc467091512)

[Skill-Based Reputarchy 35](#_Toc467091513)

[VerityDao 35](#_Toc467091514)

[Making a Profit 36](#_Toc467091515)

[VerityDAO Governance Actions 36](#_Toc467091516)

[Collaboration Layer 37](#_Toc467091517)

[Agents 37](#_Toc467091518)

[Content 38](#_Toc467091519)

[Communities 39](#_Toc467091520)

[Creating a Community 39](#_Toc467091521)

[Community Governance Actions 39](#_Toc467091522)

[Applications Layer 40](#_Toc467091523)

[Example Applications 41](#_Toc467091524)

[Decentralized Applications 41](#_Toc467091525)

[Rating and Grading 41](#_Toc467091526)

[Collaborative Crowdsourcing 41](#_Toc467091527)

[Portable Reputation 42](#_Toc467091528)

[Decision Making 42](#_Toc467091529)

[Forecasting 42](#_Toc467091530)

[Machine Learning Augmentation 42](#_Toc467091531)

[Conclusion 42](#_Toc467091532)

[Bibliography 44](#_Toc467091533)

[Appendix A – Mathematical Building Blocks 46](#_Toc467091534)

[Utility Functions 46](#_Toc467091535)

[Scoring Rules and Divergence Measures 46](#_Toc467091536)

[Pooling and Extremizing Algorithm 47](#_Toc467091537)

[Monte-Carlo Simulation 47](#_Toc467091538)

[Appendix B – Math Formulas and Algorithms 48](#_Toc467091539)

[B.1 – EigenTrust Algorithm 48](#_Toc467091540)

[B.2 – EigenTrust++ Algorithm 49](#_Toc467091541)

[B.3 – Relative Rank Algorithm 50](#_Toc467091542)

[B.4 - Value Rank Algorithm 50](#_Toc467091543)

[B.5 – Affinity Score Algorithm 50](#_Toc467091544)

[B.6 – Token Transfer Algorithm 50](#_Toc467091545)

[B.7 - Token Transformation Algorithm 51](#_Toc467091546)

[B.8 – Influence Limiter Algorithm 51](#_Toc467091547)

[Appendix C – Technical Details and Challenges 52](#_Toc467091548)

[Verity Architecture Flow Diagram 52](#_Toc467091549)

[Expensive Computations 52](#_Toc467091550)

[State Channels 52](#_Toc467091551)

[Interactive Verification 53](#_Toc467091552)

[Gas Costs 54](#_Toc467091553)

[Short Term – Market Selection and Foundation Subsidies 54](#_Toc467091554)

[Long-Term – Gas Automation and Website Subsidies 54](#_Toc467091555)

[Randomness 55](#_Toc467091556)

# Note

This document represents Verity's technical approach to reputation, governance and collaboration. The intent is to show an approach that emphasizes a building block like structure that allows for flexible, extensible pieces that can be snapped together in powerful ways that truly solve some of the long standing problems with reputation and collaboration in both decentralized and centralized contexts.

This document is not intended to show Verity's roadmap, go-to-market strategy, competitive analysis, or market opportunity. For that, please see our upcoming open source business plan, which will go deeper into these issues.

# Special Thanks

Special thanks to:

Matt Chwierut, Drew Blount, Thai Wood, Rob Caraway, George Li, David Weiner, Jonas Goldenberg, Karen Weiner, and Kate Settle. This paper would not have been possible without you.

Verity White Paper

A decentralized governance and reputation system for building verifiable, trustworthy and collaborative communities

# Abstract

We present a programmatic building block approach to enable social applications which depend on verifiable, trustworthy and collaborative communities. By layering reputation, governance and collaboration building blocks together, we show how our platform can be used in a variety of situations to model most community dynamics and cooperative goal oriented activities.

Throughout these layers, we present new reputation and governance mechanisms, that express community characteristics, in terms of agents' values, skills, and contributions. These reputation and governance metrics target and solve many of the underlying problems that have plagued internet communities in the past, and can be shared and transferred between communities, allowing for universal reputation metrics. These universal metrics can also incentivize cooperation and competition in a crowdsourcing setting, creating a task marketplace and rewarding those who offer the most value.

By putting these powerful tools in the hands of a user-owned autonomous organization, the Verity platform aims to fundamentally redefine trust and collaboration on the internet. By creating widely used tools that can interoperate in different apps and websites, Verity aims to put power back in the hands of the users, allowing users and applications to work together towards shared values and goals that they care about, and share in the profit the platform earns.

# Background

## Crowdsourcing

The idea of paying a large group of diverse people to solve a problem has been around for hundreds of years. In 1714, the British government offered £20,000 (equivalent to ~$3,500,000 today) to the first person that could create a method to determine accurate longitude within half a degree (Nesta, 2016). This was one in a long line of instances where governments used crowdsourcing contests to spur creativity.

## Wisdom of the Crowds

Another large leap regarding crowdsourcing was made in 1907, by Francis Galton. He realized that by taking the median guess of an ox’s weight among a diverse crowd of guessers, he could arrive at a number very close to the true value of the ox’s weight(Galton, 1907). This marked the beginning of the realization that crowdsourcing could be used not just for creativity, but for accuracy as well. This idea came to be known as the wisdom of the crowd. Today, the concept of crowdsourcing and wisdom of the crowd is used to power some of the most popular websites on the internet, including Wikipedia, Reddit, and StackOverflow.

## Internet Reputation

In combination with the rise in crowdsourcing, there has been a surge of interest in internet reputation metrics. One early example was eBay’s reputation system, which combined positive and negative feedback. eBay moved to show percent negative feedback in 2003(Resnick, Zeckhauser, Swanson, & Lockwood, 2006), a tacit recognition that their reputation system was having unintended consequences. Similar realizations ultimately led to the development of several new reputation algorithms designed to resist manipulation, with one of the most popular, EigenTrust, being cited over 4,000 times(Google Scholar Search, 2016).

## Combining Reputation and Crowdsourcing

As reputation and crowdsourcing began to evolve in tandem, game theorists began wondering how they could be used together to elicit better outcomes from the crowd. In 2011, the United States government agency IARPA created the Aggregative Contingent Estimation project (ACE) to answer such questions(Aggregative Contingent Estimation, 2016). One particular participant in ACE, the Good Judgement Project, was able to achieve spectacular results, using civilians to outperform CIA analysts by 30% (Spiegel, 2016).

## Ethereum

In late 2015, Ethereum launched, a platform for creating smart contracts on the blockchain. For the first time, decentralized applications (dapp) became almost as easy to build as standard web applications, and money was as almost as easy to program as any other piece of data. This represented a way to measure and store metrics such as expertise and reputation that are immutably linked to identities, a feature previously found only in real life reputation. The programmable money aspect also provided intriguing possibilities for incentivized crowdsourcing.

While Ethereum represents a huge paradigm shift in the power to create applications that don't have a centralized failure point, many have pointed out that there are still fundamental limitations to the type of decentralization that can be achieved(Sztorc, 2016). In particular, critics have noted that any connection to external data feeds, and any logic that requires human judgment, cannot be decentralized purely with smart contracts. Critics theorize that data must be decentralized, or self-generated inside the decentralized application itself. In addition to crowdsourcing and reputation uses, Verity precisely fills this hole for general decentralized applications, which depend on verifiable, trustworthy and collaborative communities.

# Summary of the Verity Platform

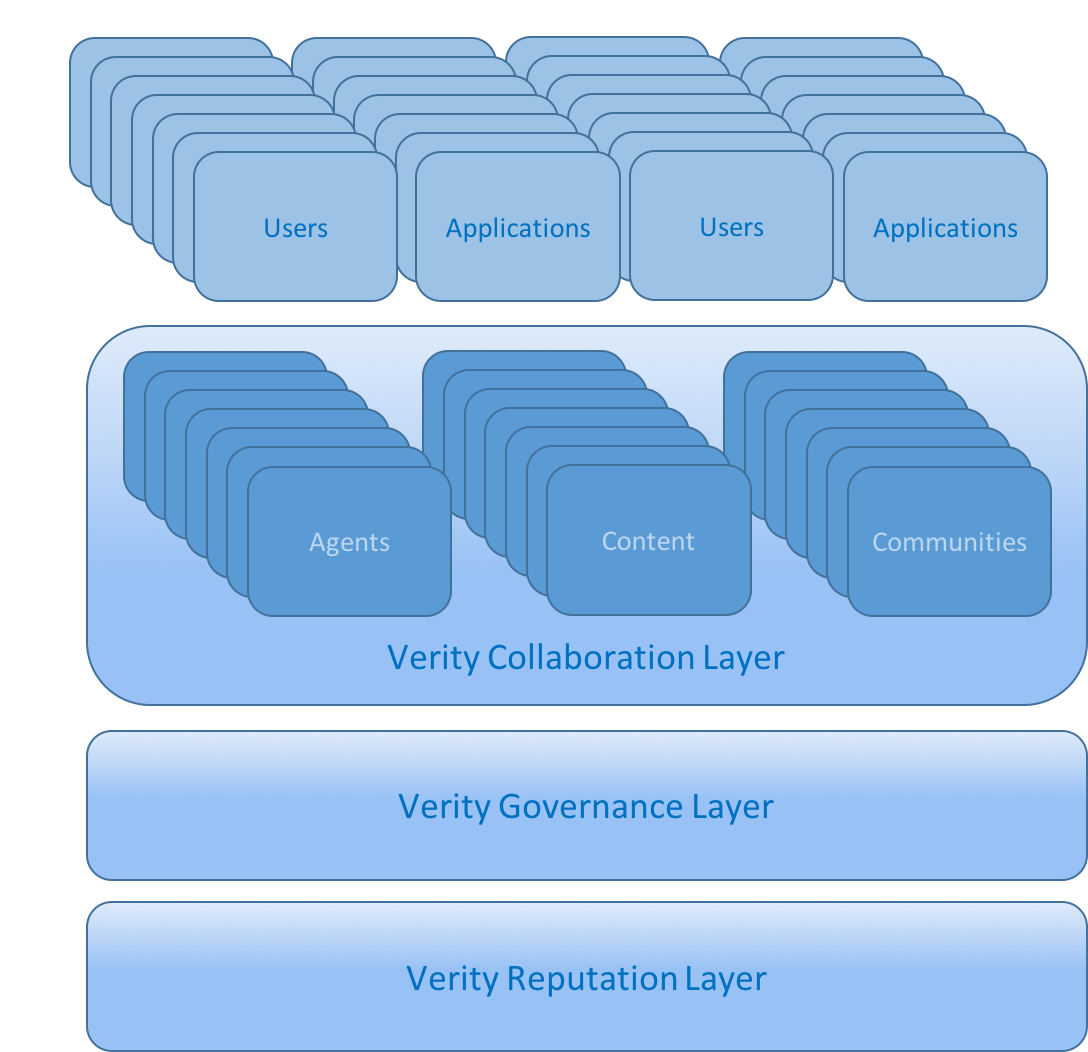


Figure - Verity Platform

The Verity platform provides a set of open source building blocks that enable the development and operation of highly collaborative applications. Developers may utilize Verity's reputation, governance and collaboration solutions together to build highly collaborative applications that seamlessly integrate with every other application on the Verity stack.

This means that the Verity platform serves as a sort of “reputation and governance” glue for collaborative applications, giving them a common language, consistent experience, easy way to communicate, ability to share users and the power to leverage common reputation systems.

## Reputation Layer

Verity's **reputation layer** consists of a general **reputation standard** for describing, accessing, and updating reputation metrics for both agents and content. This allows for an ecosystem of plug-and-play reputation metrics to be developed, which can be immediately integrated into Verity' communities.

On top of this standard, Verity builds its three foundational **reputation metrics** that can rate agents on the values they exhibit, the skills they possess, and the contributions they make to their community.

Values are ranked through a web-of-trust system, and are assumed to be transitive meaning those whom exhibit values are assumed to be better at judging those same values in others.

Skills are ranked by a system of competition where agents who perform well earn reputation from those who perform poorly. In this way, skills based-reputation measures the relative skill-set of an agent compared to other agents.

Finally, contributions are ranked by measuring every agent’s contribution to a pre-defined community goal. In this way, contribution scores are a measure of how much an agent has helped a specific community.

## Governance Layer

Verity's **governance layer** allows communities to make decisions and take action based on those communities' own governance rules. It implements a **governance standard** that allows for a flexible system of roles, permissions, and proposals. By layering these systems together with the power of smart-contract code, an ecosystem of plug-and-play governance tools can flourish.

On top of this governance standard, Verity creates three entirely new forms of governance based around Verity's three reputation metrics. The first, **value-based hierarchy**, allows governance roles to be tied to the values an agent is rated with in values-based reputation. The second, **contribution-based voting**, allows agents to have votes commensurate to the contribution they have given to the community, as measured by contribution-based reputation. The final governance system is **skill-based reputarchy**, which gives agents influence commensurate to their ability to make good decisions, as measured by skill-based reputation.

## Collaboration Layer

The **collaboration layer** in Verity is built on the notion of **agents**, **content** and **communities**. **Agents** are any users, groups, or autonomous actors that can perform actions within the system. A **community** in Verity is a group of agents that share a common purpose, vision, or goal. **Content** is anything in the Verity protocol that can be rated or acted upon, but which cannot itself act.

By grouping agents into communities, shared intent causes communities to self-organize talent, resources and actions around economic and social incentives. Verity uses governance rules and reputation scoring metrics to preserve community intent and incentivize actions in alignment with that intent.

Checks and balances in the Verity protocol encourage community collaboration consistent with agreed-upon governance. Community core values and goals can be explicitly defined, but norms for what those values and goals mean are flexible and moderated by the Verity value score algorithm. Community members share in rewards and common good to the extent that they are recognized by their peers through reputation and can be rewarded percentages of income based on outcomes derived from contests and reputation score.

Verity is unique in its ability for agents to transfer reputation between communities while maintaining their reputation. This ability is what allows Verity to be a global measure of reputation, instead of a series of stand-alone or one-off metrics such as Reddit karma or STEEM power.

## Application Layer

The **application layer** in Verity refers to the ecosystem of mobile apps, web apps, and decentralized apps that implement service calls to Verity's protocols. Verity's open source standards ensure that every innovation built on the platform is usable by all applications, and Verity's Ethereum backbone provides a common data store for reputation, allowing reputation information about agents and content to be shared and transferred between applications. This gives Verity's application layer a unique network effect which can grow through any individual application gaining more users, or through more applications coming into the system.

Users and applications interact equally and consistently within Verity's ecosystem, while benefiting from diverse solutions regarding identity, privacy, and access. Choice of governance settings integrates preferences into the Verity protocol, allowing seamless interaction between a variety of different applications, all with different needs and business models to coexist, scale and flourish.

Verity’s initial governance and reputation systems provide user owned solutions designed to be attractive to users of applications while serving the needs of the application creator as well. This allows users and applications to work together cooperatively with aligned incentives and goals to deploy governance settings that best meet their joint needs. This is in contrast to user and application terms and conditions of the past that put the needs of users and applications at odds.

Since the Verity protocol translates everyday user-application behavior into verifiable reputation metrics, the burden to manage “sensitive” metrics data is offloaded to the protocol itself. This allows applications to focus on core functions related to their specific purpose, yet access metrics data when they need it.

## An Example

As a concrete example of how Verity could be used, let’s take the problem of finding agents who can write and audit smart contract code. In the Ethereum ecosystem, the hack of The DAO exposed this as an unsolved problem. Using Verity, one would create a “smart contract security” community. This community’s reputation metrics and contests would then serve as the backend for a number of important decentralized applications (dapps) needed in the ecosystem such as:

* A dapp for hiring qualified people to code your custom smart contract.
* A dapp for paid, crowdsourced smart contract security audits from qualified professionals
* A dapp for decentralized reputation-based commit access to open source smart contract projects
* A dapp to get advice from other smart contract coders on best practices (with the best advice from the most knowledgeable coders being ranked the highest)

Because each of these dapps would share the same reputation metrics and governance tools, each one would in turn improve the accuracy and utility of the other dapps, and the ecosystem as a whole would benefit from new dapps built on the platform in a way not seen with siloed reputation metrics.

# Reputation Layer

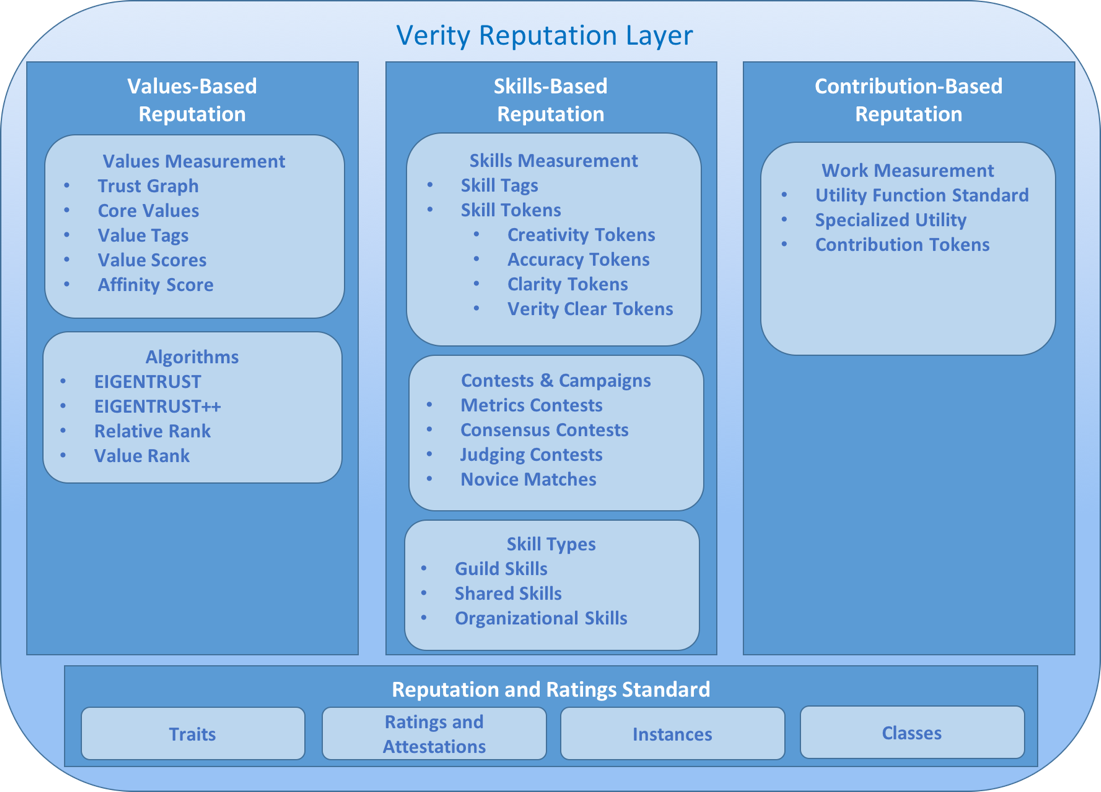


Figure – Reputation Layer

Verity’s **reputation layer** provides a general purpose **reputation and ratings standard**, which acts as a simple standard interface for accessing and understanding reputation and ratings, similar to the ERC20 token standard(Vogelsteller, 2016). On top of this standard, we introduce three **reputation metrics** that represent the three types of reputation typically used in real life scenarios. One is for **values-based reputation**, with which an agent's values in the eyes of others are rated. The second is **skills-based reputation,** with which an agent's skills are ranked based on the output of their skills such as the ability to write forum posts or debug code. Finally, we measure **contribution-based reputation**, which measures an agent’s contributions to a community or organization over time. Throughout this document a number of references are made to mathematical building blocks, formulas and technical details. For more information, please see the appendixes.

## Reputation and Ratings Standard

Verity’s **reputation and ratings standard** is a set of common functions that allow smart contracts to implement ratings and reputation metricsin a consistent way, thus allowing them to keep the same interface for their smart contract even if they change reputation metrics.

The standard is simple, and is built on top of Verity’s notion of communities. It gives several functions to return information about the reputation or rating metric such as what it should be called, the type of reputation it is (categorical or numerical), and how it should be displayed (limits, units, etc.).

### Reputation Traits

Every reputation contract has **traits** that define the use cases for that metric.

* A reputation contract can be **numerical** (meaning people are ranked in that reputation according to some number) or **categorical** (meaning that people are put into discrete buckets).
* Numerical reputation has an optional **minimum** and **maximum** trait.
* Reputation can also be used to rank **content** (such as ranking restaurants) or **agents** (such as deciding how trustworthy someone is), or both.

### Ratings and Attestations

Many reputation metrics require agents to attest to or rate either content, or other agents. Verity's rating standard has a set of common functions to allow agents of a community to create **attestations and ratings**. Crucially, an implementation of the ratings standard is separate from an implementation of the reputation standard. This allows different reputation types to be built on the same base data.

### Reputation INstances

Every reputation metric has an optional instance system, which allows the same reputation system, in the same community, with the same treats, be used to measure different reputation metrics. For instance, a community could have one instance of "karma" that measures regular users, and another that measures moderators. This system allows a community to use the same mathematical reputation rules to measure a variety of different things, and allows new measurements using the same reputation rules to be created at will.

### Reputation Classes

Our reputation standard is set up such that one can create different classes of the same type of reputation. These reputation classes all share the same basic building blocks, but apply to different communities, may have different traits, and do not share the same instances. This system allows for two communities to use the same *type* of reputation without sharing reputation between them.

## Values-Based Reputation

Existing ratings and reputation systems built into communities such as Reddit are able to classify the type of content that the community likes, but are not able to identify the type of people that should hold influence in their community according to community values. This means that any open community that uses such a content centric system will be driven by populist values and conflicts, often causing communities to be held hostage to random storms of hostile dialog with the only recourse being some form of centralized censorship.

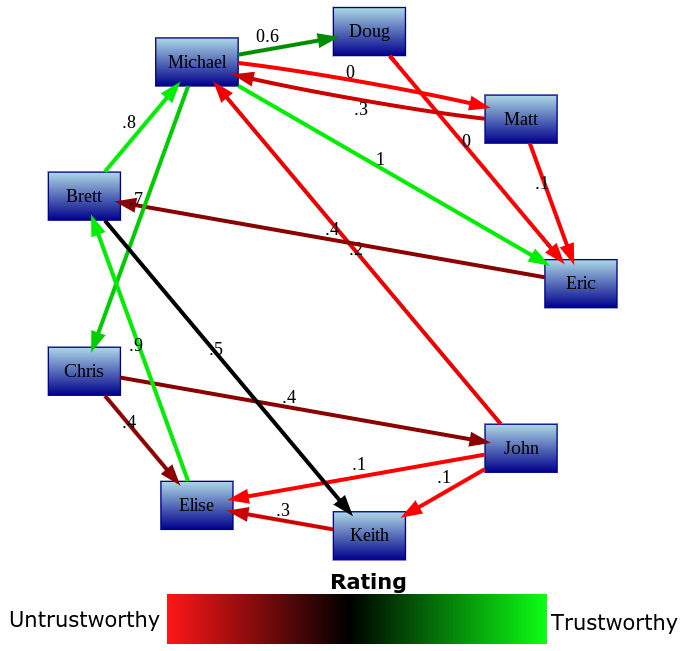


Figure - Example Trust Graph

Verity’s **values-based reputation** expresses fundamental characteristics of reputation that records and maps human values into a **trust graph**. By allowing the community to define values, it allows for maintaining community integrity without needing a centralized moderator. Using values-based reputation, communities can choose **core values** that they want their members to hold. Core values can influence various aspects of community involvement, such as ratings or stake in community decisions.

Each agent in the system is ranked with an **affinity score**, which shows how aligned they are with a communities values. Agents can gain influence in the community if they are in alignment with the values of that community. In this way, values-based reputation eliminates trolling and provocative behavior and can enable application of moral decision-making, such as deciding what content should not be allowed on a platform.

### Value Tags

**Value tags** are simple text representations of the value that will be used in a values-based reputation rating. Each tag has a “maximum” and “minimum” description that describes the value and its reverse. For instance, the honesty tag may have the maximum of “honest” and the minimum of “dishonest”. Each tag will be registered in a global registry along with a URI to a longer free-form description of the value. Long form descriptions can be fuzzy and general, or precise enough to be used in a contractual agreement. For the purpose of Verity reputation calculation, tags and descriptions are arbitrary and only have meaning in the context of the reputation trust graph of the community. The Verity algorithms will work with any language, jargon, or concept.

Tags and descriptions can be created and registered at will, or existing tags in the global registry can be reused if they are a good fit for a community. While agents can rate agents and give personalized local scores to any agent for any value, global value scores are calculated only for values which are tracked by some community or group of communities, and are interpreted through the lens of that community's understanding of the value.

### Value Scores

An agent can have a **value score** for eve (Kamvar, Schlosser, & Garcia-Molin, 2003)ry value tag/community pair that exists on the platform. They are a permanent record for the account that earns them, and hold a value between -1 and 1. Value scores above 0 can generally be seen as having that value, while value scores below 0 can be seen as the reverse (as defined in the value tag metadata). Intermediate values can be seen as degrees of affinity with that value.

#### How Value Scores Work

Value scores represent specific qualities that agents of the system hold. They're computed by combining versions of **Relative Rank**(Traupman, 2007)and **EigenTrust++**(Fan, Liu, & Li, 2012), two Sybil-attack resistant versions of the **EigenTrust** (Kamvar, Schlosser, & Garcia-Molin, 2003) algorithm that normalize based on the number of ratings each agent has been given, and incorporate the structure of feedback that agents get.

##### EigenTrust

The **EigenTrust** algorithm is based on the notion of transitive trust: A peer will trust agents trusted by those agents it trusts (and so on). The EigenTrust algorithm calculates a local trust value by taking all positive interactions, subtracting all negative interactions, and then propagating this trust transitively along all agents.

The authors of the EigenTrust paper note that for networks with a sufficiently large set of agents, the aggregated local trust vector *t* always converges to the same number, regardless of which agent *i* it’s calculated from. This means that the trust vector *t* represents a global notion of trust that the network places on any given agent.

###### Start Sets

The authors also note that there are a few problems with the simple algorithm above. Firstly, malicious agents can create networks specifically designed to increase the score count between each other, by “trapping” the probabilistic crawler mentioned above in a web of trust links.

Secondly, if an agent has not rated any peers (or has rated all peers negatively) cij will be undefined, making the algorithm impossible to compute.

They solve both these problems with the notion of a trusted “**start set**.” This start set represents a set of trustworthy peers that have not been compromised. To remove the chance of the algorithm being trapped in a Sybil compromised network, they make sure that every peer has at least some small amount of trust allocated towards the start set, such that on any given step the algorithm can exit the malicious network by returning to the start set.

If an agent has not rated any peers in a positive way, then that peer is treated as having implicitly given its trust to the start set, thus avoiding dead ends in the network. For our purposes, we distribute trust evenly among every agent in the start set.

The algorithm for calculating EigenTrust is shown in **B.1 – EigenTrust Algorithm**.

##### EigenTrust++

**EigenTrust++** suggests three ways to increase the attack resilience above classic EigenTrust. Firstly, it adds the concept of “feedback similarity” to its trust propagation neutralizing a class of attacks that works by acting honestly, while rating dishonestly (in order to increase your reputation relative to peers who both rate and act honestly). Secondly, it incorporates information about how many feedbacks a peer has received, allowing for peers with lots of negative feedback to be treated differently than peers with low amounts of feedback. Thirdly, it creates thresholds for trust propagation, reducing the possibility that dishonest collectives can pass on reputation to honest agents, and vice versa.

For the purposes of this paper, we utilize EigenTrust++’s feedback similarity rating, and linear threshold, but don’t use its incorporation of feedback number. This is because we include a different normalization procedure based on feedback number, described below in the section on Relative Rank.

###### Feedback Similarity

EigenTrust++ notes that a peer can maliciously attack the network by always acting honestly when interacting with high reputation peers, but interacting dishonestly in other situations. It solves this problem by creating a “feedback similarity” metric which allows honest agents to detect this type of behavior, and propagate less trust to agents that engage in it. It has an added bonus for our algorithm, as it captures the notion of subjective values – an agent will trust other agents that see the value in the same way that it does.

###### Trust Propagation

While EigenTrust counts both feedback similarity as well as trust level equally for the score itself, it recommends that you weight feedback similarity higher when determining trust propagation. It recommends some percentage β that you should weight the similarity in proportion to the propagated trust score.

The authors of the EigenTrust paper suggest that β should equal .85.

###### Propagation Thresholds

EigenTrust++ also recommends propagation thresholds, after which trust won’t be propagated at all to a new agent. While EigenTrust’s start sets do minimize the damage of malicious collectives, propagation thresholds aim to go a step further and actually penalize these collectives and other bad actors. EigenTrust++ treats this threshold probabalistically, to strike a good balance between penalizing clearly negative agents, while allowing agents whom were unfairly given negative reputations to still participate in the system.

The algorithm for EigenTrust++ is shown in **B.2 – EigenTrust++ Algorithm**.

##### Relative Rank

**Relative Rank** is an algorithm that seeks to add additional Sybil-resistance to the EigenTrust algorithm, while at the same time making it more suitable for peer-to-peer markets. By transforming EigenTrust’s arbitrarily high trust vectors into a normalized value, Relative Rank creates a clear decision procedure to determine if a peer should be trusted or not within an interaction. The normalization procedure also seeks to include negative feedback, in order to separate dishonest agents from agents whom have simply not been ranked. In order to create this procedure, Traupman first analyzed the behavior of EigenTrust in marketplaces, then tried to determine a clear threshold in the determination of whether an agent was trustworthy or untrustworthy. We use the same procedure but apply it to EigenTrust++.

The algorithm for calculating Relative Rank is shown in **B.3 – Relative Rank Algorithm.**

##### Value Rank

**Value Rank**, our algorithm for calculating the values that an agent holds, makes two minor changes to the original relative rank algorithm. It makes local ratings more granular, and it allows for multiple start sets.

The algorithm for calculating Value Rank is shown in **B.4 - Value Rank** Algorithm.

###### Granular Local Ratings

In the original relative rank algorithm, feedback is binary, and interactions could be rated only positive (plus one) or negative (minus one). While this makes sense for the original implementation of EigenTrust, in which a peer either gave the correct data or did not, it does not allow for the nuance that comes with arbitrary values, such as deciding the level of kindness that an agent showed. In relative rank, this also means that there is less distinction between agent ratings, because of the high correlation between *r* and *k*. For this reason, agent ratings are given as a decimal value between positive one and negative one, allowing for more granularity in every interaction. By multiplying the final relative rank by 2 and subtracting 1, agents can also get a clear intuition for what a relative rank score means - an agent rating of *-0.3* means the same thing as a relative rank score of *-0.3.*

###### Multiple Start Sets

In the original relative rank, a single start set is used. A related algorithm is also given called RAW which allows for personalized start sets. In value rank, multiple start sets can be created, one for every community that ranks that value as their core value. This captures the notion of values as relative interpretations that exist within community context, while still striking a good balance with computational cost.

#### Community Level Value Scores

The Relative Rank algorithm used to calculate value scores has the ability to calculate different value scores based on what group of initial "trusted agents" it calculates from. This is ideal for growing out a global trust graph while allowing agents to build value scores faster in the community they're a part of. The ultimate goal for value-based reputation in Verity is to build a global **trust graph** that spans a diverse network of communities and scales to planetary levels.

##### Community Core Values

In Verity, each community can choose to create its own set of **core values**, and for each of those core values can define one or more community members they consider paragons of that value. This becomes the initial start set from which the trust graph grows. As new members on-board and participate in Novice Matches they are rated by the existing members thereby populating new agents in the trust graph.

For more on Novice Matches, see **Onboarding Agents in Verity,**

##### Affinity Score

As agents begin to get rated in a community’s core values, we begin to get a sense of their affinity for the community. We can mathematically represent this affinity by averaging their score along all core values.

We call this average of all community core values for a specific agent that agent’s **affinity score.** The affinity score is used throughout the Verity platform to make sure that agents who have influence in the community are in alignment with the values of that community.

The algorithm for calculating Affinity Score is shown in **B.5 – Affinity Score Algorithm.**

#### Value Scores in Action

Let's imagine how value scores might be used on a site like reddit, with many different communities that can interact with each other.

Firstly, each community, would choose a set of core values, which represented the absolute minimum for being allowed to interact with that community. A simple example might just be "decency", which any number of communities might adopt as simply the price of admission for participating in their community.

For each set of core values the community would choose a start set of people they considered paragons of that value. In the case of decency, the community would nominate people whom were universally liked and cool-headed. These people in turn would begin to rate other people in the trust graph, and this would be combined with the already existing trust graph to get a clear idea of what each member considered decent. This would act as a filter, keeping people in a sandboxed version of the discussion forums until they could prove they were decent.

Secondly, aside from the "price of admission" core values, additional value scores could be used as weights for voting and posting, allowing for community norms to be upheld without restricting access. For instance, an "existentialist jokes" community might have a particular brand of humor that it wanted to preserve even as the community grew larger. This community could weight it's votes and new posts by it's own humor ranking, allowing everyone to participate while still preserving the type of humor that it valued.

## Skills-Based Reputation

**Skills-based reputation** is the second foundational type of reputation. While values-based reputation answers the question “What type of person are you?” skills-based reputation answers the question “What are you good at?”

Existing methods of measuring skills typically fall into one of three categories:

* Standardized assessments that try to measure different aspects of an agents’ performance
* A record of the education a person has received
* A fuzzy assessment based on an agent’s accomplishments or portfolio.

In Verity, skill measurements are based on an agent’s ability to use those skills to fulfill the goals of the people they’re working for, relative to other agents. While this definition may still seem fuzzy, Verity’s power lies in its ability to define these goals in a precise mathematical way, and to keep an immutable history of an agent's performance on these goals that can’t be tampered with. These innovations allow Verity to measure skills in a precise way that people looking to work with those agents actually care about.

### Skill Tags

**Skill tags** are simple text representations of the different skills that agents can hold, along with a URI to a long form description of the skill. Skill tags can be general and fuzzy, or specialized and precise. Skill tags can be created at will, or existing skills can be reused if they fit the task at hand.

While any agent can claim to have a skill, skill ratings only exist in the context of a community or group of communities. This ability to share skill ratings among groups of communities allows for the possibility of widely used skill measures that can become defacto standards.

### Skill Tokens

Skills in Verity are measured by the amount of **skill tokens** that an agent holds. These tokens are typically “colored tokens” that represent specific skills in communities.

There is also a special type of token called Verity clear tokens that don’t yet represent a specific skill in a specific community. This type of token is the only type of token that can be used to create new skills within a community. The following four types of skill tokens exist, which correspond to different skills related to a specific skill tag.

1. You can actually use the skill to solve problems. This will earn you **creativity tokens**.
2. You can critique agents’ use of the skill, this will earn you **clarity tokens**.
3. You can grade agents’ performance against some standard, this will earn you **accuracy tokens**.
4. You can support the platform through buying tokens in an initial sale of them. This will earn you **Verity clear tokens.**

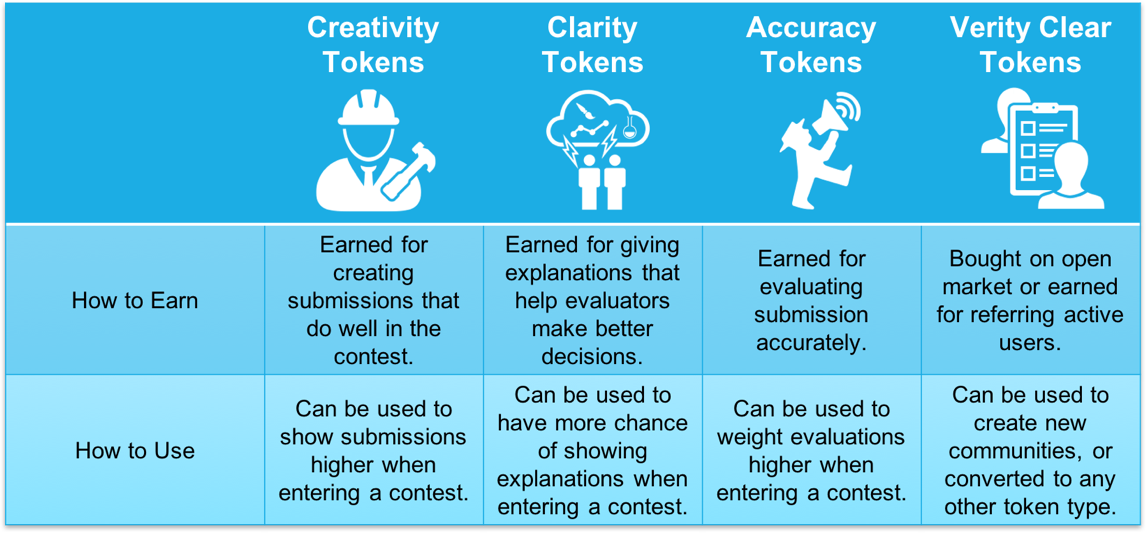


Figure - Verity Token Types

#### Creativity Tokens

To earn creativity tokens one must generate new content. If that content is seen as valuable in relation to other content, one will earn more creativity tokens. Submissions don't have to be in the form of text. An idea could just as easily be presented in the form of code, blueprints, or mockups - allowing for creativity tokens to be used for arbitrary crowdsourcing applications.

#### Accuracy Tokens

To earn accuracy tokens, one must evaluate how well submissions meet the variables that go into the utility function specified by the client. Evaluators provide probability estimates among all these variables for the options generated by creatives. This allows the client to make the decision-theoretically optimal decision.

#### Clarity Tokens

To earn clarity tokens, one must explain the pros and cons of different submissions. You can think of these pros and cons as analogous to comments on traditional crowdsourcing sites. However, these explanations do not just have to be in text, and can be in arbitrary media depending on the application. While the client gets to see all explanations by every critic, the explanations are only shown to evaluators and creators on a probabilistic basis to enable linear regression analysis of the explanation’s impact. This allows clarity tokens to get redistributed to those who most help the predictors make accurate predictions, and who help the creators make valuable creations.

#### Accuracy and Clarity Tokens as Forecasting

When the skills of grading and critiquing are used before the outcome of an event is known (example: a battle plan is critiqued and graded before the battle), they correspond to the ability to both understand and predict the future.

This type of crowdsourcing-based forecasting combined with reputation has been shown to outcompete established agents like CIA analysts by 30%(Spiegel, 2016), and is comparable with the accuracy of prediction markets when the proper algorithms are used(Atanasov, et al., 2015).

It is this ranking of forecasting ability that gives these meta-skill reputation tokens (clarity tokens and accuracy tokens) equal value to the primary skill-based token creativity tokens.

#### Verity Clear Tokens

Verity Clear tokens are a Verity Tokens that have not yet been converted to tokens marked for a particular skill-tag, community, and token type (Colored Verity). Verity Clear can be sold as assets for purposes of fundraising for the platform. Verity Clear tokens are special in that they are not used as a measure of reputation in themselves. Verity Clear can be transformed by any agent at any time, into any of the tokens above, with any skill tag, in any community- subject to that agent's expected token score for that token type. They are also the only tokens that can be used to originally create a new measure for a skill tag within a community.

#### Portable Skill Tokens

One of the current problems with internet based reputation is that accounts can be easily sold if not tied to strong identities, thus making it impossible to know if reputation was earned or simply purchased.

Verity solves this problem by allowing the tokens used for reputation to be transformed into portable, sellable tokens, but distinguishing between tokens that are factored into reputation (activated tokens) and tokens which are not factored into reputation (deactivated tokens). The way it does this is by incentivizing the sale of or transformation of reputation tokens only to people who deserve those tokens, and deactivating the tokens if sold to others. By doing this, it gives people honest ways to profit from reputation tokens they no longer have a use for.

In Verity, we measure similarity by assuming that communities are similar if people that have more of a token in community *a* also have more of a token in community *b.* This relationship is easily calculated using an asymmetric measurement of Pearson Correlation, which shows if there is a linear correlation between a and b. This score is then used to calculate the exchange rate.

##### Exchange Rate

In a typical currency, the exchange rate is determined by relative demand for two different currencies. Verity, being a reputation token, has an entirely different notion of exchange rate that mimics how reputation works in real life. Verity creates a simple linear equation that shows how many of a certain reputation token we'd expect them to have, given how many tokens they have of the reputation they're exchanging from, and then uses this equation to create an exchange rate when converting from one to the other. The equation is done by doing a linear regression of all agents who hold tokens in either community.

##### Token Deactivation

Token deactivation serves as a mechanism to make it hard for people to use their tokens to claim reputation in an expertise they don’t have. Inactive tokens cannot be used in contests. The only way for tokens to get reactivated is through earning the same type of tokens in contests. For every token of the same type earned, an inactive token is reactivated. The flipside is that for every token of the same type lost within a contest, you lose a corresponding deactivated token. These lost deactivated tokens are then given back to the community to be distributed through novice matches. What this means is that owning deactivated tokens causes you to be able to both win and lose tokens twice as fast. This encourages agents to only be willing to hold deactivated tokens that they know they have expertise in.

When calculating how many tokens should be deactivated during the sale or transfer of a token, the agent’s entire token portfolio, along with the exchange rate, is used to calculate an “expected tokens” value for the community they’re moving into. Every token an agent holds in all communities (except the community for which tokens are being bought) is multiplied by its percent similarity with the community token being bought. These numbers are then averaged together, and weighted by affinity to the target community, to calculate the ‘expected tokens’ that the agent should have in that community. Any tokens purchased that exceed this number are deactivated.

This discourages whales in a community from buying or transforming tokens to cement their advantage, because they are likely already above their allotment of expected tokens. It also discourages non-agents from buying influence in contests, because their expected tokens are likely very low. What it encourages is agents from similar communities who don’t want to sacrifice tokens through transformation to buy tokens from communities they should do well in. This transfer of crypto-reputation to people who deserve that reputation in real life is the exact behavior we’d like to encourage.

The algorithm for calculating Token Transfer is shown in **B.6 – Token Transfer Algorithm**.

###### Transforming Tokens Using Similarity Score

When transforming tokens from one community to another (or one contest type to another), a slightly different procedure is used. The similarity between the two communities is used to figure out how many tokens are deactivated *in each transfer*.

This procedure holds up to the point at which the agent reaches their expected tokens for a community. After that point, the agent is no longer allowed to transform tokens into that community.

This procedure preserves the incentives of transferred tokens, and adds the additional incentive to transform tokens into tokens which represent similar skills. This allows skill tokens to take on a price related the value of that skill in the marketplace, instead of being merely an average of every skill in the market.

The algorithm for calculating Token Transformation is shown in **B.7 - Token Transformation Algorithm.**

###### Portable Tokens in Action

How might transferring and transforming tokens work in the real world? Let's imagine how this situation might play out with Kerry, a graphic designer who has earned her living for a number of years in a community centered around graphic design, and would like to expand her skills to the web design realm. Without Verity, she would have had to rebuild her reputation from scratch losing time and money in the process. With Verity’s reputation metrics automation, she can quickly and easily gain the reputation she deserves in a new skillset through token transfer and transformation. She has 500 tokens in the graphic design community, and her expected tokens in the web design community are 300. The communities have a similarity score of ,6.

Kerry knows from her experience with graphic design that it could take her 6-months to a year to gain the full 300 tokens in the web design community. She does a quick estimate in her head, and realizes that by having the skill-tokens immediately, she can make about $10,000 more over the course of the year. Verity Clear tokens are selling for about $20 on the open market, so Kerry decides that with her savings, she can buy about 150 Verity Clear tokens, for a total cost of about $3,000.

The remaining 150 she transforms from her graphic design tokens, 90 of which are activated immediately, and the remaining 60 of which will be activated one by one for each token she earns in graphic design. Kerry can now immediately get recognized and hired for better jobs n the web design community, and recoups her investment in Verity Clear tokens by the second quarter in which she's working in that community, presenting a significant improvement over manual siloed reputation systems of the past.

### Verity Campaigns and Contests

#### Campaigns

A **campaign** in Verity refers to a single unified event in which agents compete to deliver a multi-faceted solution for the Client. Every campaign is paired with a **utility function,** which defines a number of different inputs, and outputs a number that measures the quality of each solution.

Each campaign can optionally be broken down into a number of **contests**, one for each input to the utility function. In this way, different inputs to the utility function can be measured by different agents with different rules, allowing for specialization within the campaign and allowing agents to work together to maximize utility for the client.

##### Campaigns in Action

For instance, a reddit-like site that aims to show agents the best agent created content might have several criteria that it wants to consider when ranking posts for agents:

* An objective criterion, such as how many pageviews the post gets.
* The will of the agents, expressed by upvotes and downvotes.
* An agent agent’s tastes, based on their past history of upvotes and downvotes.

Each expert can then compete in a variety of contests, depending on their expertise

* They can choose to create content that they believe will perform well along all metrics
* They can predict or explain how many pageviews the post will get, and get tokens for being correct
* They can upvote or comment content that they think will be upvoted by others and get tokens for being early.
* They can choose to recommend specific content they upvote to similar agents, and get tokens if those similar agents end up liking it.

In Verity, the only way to earn skills-based reputation tokens is by entering into a contest. A contest is a community level event in which an agent competes against other agents in the community to create the best content.

Contests can be repeating, such as trying to create the best content for the day in a Reddit-like website. They can also be one time, such as a company making a contest where agents compete to create a new product line. Verity’s standard contest types all go through a standard process, during which agents compete with Creativity Tokens, Clarity Tokens and Accuracy Tokens.

##### Contest Standard

All contests are implemented as smart contracts that follow a predefined standard. The contest standard defines a temporary, individualized type of reputation that only exists for the duration of the contest. Using Verity's standard tokens, these contests can then be combined together into a single reputation type which can use different rules depending on the use case. This allows new crowdsourcing applications with varied needs to incorporate Verity reputation into their own unique contest types that are not covered by standard Verity contests. It also allows for various contest types to be experimented with, such as the algorithms provided by fellow crypto-reputation platforms Backfeed, Augur, Steemit, and Synereo.

#### Mathematical Building Blocks

All contest types built in to Verity use essentially the same basic mathematical building blocks. The Client provides a **utility function**, which is a mathematical representation of their preferences. Then, agents try to fulfill those preferences. Their submissions are ranked on each preference using a **probability distribution**, which is a mathematical representation of their uncertainty about how the submissions should be ranked according to the client’s criteria. These probability distributions might be over a **binary outcome,** spread over several **categorical options**. Or be trying to pinpoint a number in a **scalar outcome.** The probability distributions are combined using **pooling,** which is a mathematical tool to turn multiple probability estimates into one combined estimate, and oftentimes made more accurate with extremizing. A **monte-carlo method** is then used on these final probability distributions as they fit within the utility function. A monte-carlo method can be thought of as a guess-and-check method that a computer uses. Finally, either a **Bayesian scoring function** or a **divergence measure** is used, comparing the final values to the values each agent came up with. A Bayesian scoring function is a way to compare an agent's performance to some idealized notion of performance over time, and a divergence measure does the same thing, but compares against an ideal distribution instead of a single answer. The results of this function are used to redistribute tokens.

Please see **Appendix A – Mathematical Building Blocks** whichdefines these terms further, and provides additional resources to learn more.

#### Contest Types

There are three separate types of contests in Verity. With just these three types, almost any criteria one can think of can be included in a utility function. The three contest types are:

1. **Metrics Contests**: Contests in which agents compete against some objective criteria provided by an oracle.
2. **Consensus Contests**: Contests in which agents compete to please or agree with their peers
3. **Judging Contests**: Contests in which agents are evaluated along subjective criteria by a judge.

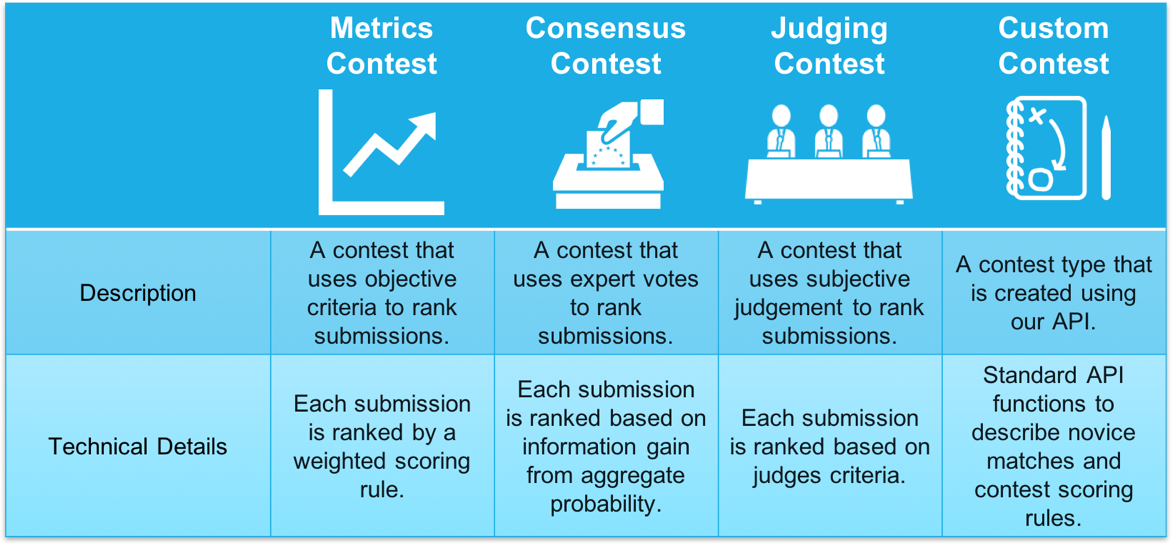


Figure - Verity Contest Types

##### Metrics Contests

Metrics contests are contests in which agents’ probabilities are ultimately judged based on real world data provided by an oracle. This oracle could be a data feed provided by a provider like Oraclize it[[1]](#footnote-2), or could even be some sort of Schelling-point oracle such as SchellingCoin (Buterin, 2014) or a Verity consensus contest. This is analogous to crowdsourcing sites like the forecasting site gjopen.com, and the data science site kaggle.com, which both use objective real-word criteria to rate their agents.

###### Metrics Contests in Action

Let’s imagine how a metrics contests could be used in our smart contract security community.

One place they could definitely be used would be in the auditing of smart contracts. For instance, a simple utility function could be defined which included the rating of the Dapp on the Dapp store in one year’s time, and the amount of bugs found in the code in one year’s time, using a clear definition of how serious something had to be to be considered a bug, and an oracle such as a Verity consensus contest.

Smart contract creators would then earn Creativity tokens by creating the most highly rated, bug free code, Smart contract critiquers could earn Clarity tokens by pointing out bugs in the code and problems with the design that would cause it to be rated poorly, and Smart Contract evaluators would predict the number of bugs and rating on the Dapp store. At the end of a year, once these values were known, reputation tokens would be distributed accordingly.

##### Consensus Contests

Consensus contests measure community agreement on a subject. They can be used as Schelling-point oracles similar to Augur’s reputation metric, to rank options against each other based on their community acceptance as in Steemit, or to gauge a communities take on intangibles such as rating how beautiful a piece of artwork is. This is analogous to crowdsourcing sites such as the tech advice site StackExchange, which use consensus based mechanisms to rank their participants.

###### Consensus Contests in Action

Firstly, one can imagine a StackExchange-like forum where programmers could discuss the creation of smart contracts. The entire site would be a daily consensus contest to ask the best question, and the questions themselves would be contests to determine the top answer. One would get Creativity tokens for asking and answering questions, Accuracy tokens for voting on them, and Clarity tokens for commenting. Instead of a single up or down vote, voting would be more like range voting, in which you could allocate however many votes to every question or answer you voted on, and the entire distribution of your votes would be interpreted as a categorical distribution, with your amount of votes indicating something akin to your probability estimate that this is the best question.

Secondly, as mentioned above, consensus contests could be used as a kind of decentralized oracle for metrics contests, using the game theoretic mechanism of Schelling Score. If you made a consensus contest for instance around how many bugs had been found in a particular contract over the past year, you’d be incentivized to count the obvious bugs that everyone else would also count, and everyone would most likely cluster around the same set of numbers at roughly the same probabilities. This allows you to have metrics contests without any worry of centralization.

##### Judging Contests

Judging contests are subjective contests. If this option is chosen, a judge ultimately rates the ideas (although voters can still help to eliminate options and guide the judges). Pre-judgement scores are also weighted by similarity between the judge and the agents, in order to show the judge the proper weighting before they make their decision(s). This is analogous to crowdsourcing sites such as the design site 99designs, which use subjective criteria to choose winners.

###### Judging Contests in Action

Let’s look at how judging contests might play out in our smart contract security community.

While security seems like a mostly objective goal, with little room for subjectivity, it’s nonetheless possible to imagine a scenario in which you might want to include a judging criterion in a contest. One example might be for decentralized commit access to an open source smart contract. While the ultimate goal might be full decentralization, it might be pertinent to start out with a long time contributor acting as judge, looking at things like coding standards and code clarity to make their judgements. Only once community members had learned the judge’s tastes (and those who didn’t eliminated from the pool) would the commit access become fully decentralized with a consensus contest.

##### Earning New Tokens Through Contests

In addition to whatever reward was posted into a contest by a client, agents also earn newly minted tokens by participating in Verity contests. These tokens are given proportionately to each contest based on how many agent tokens are in that contest, and then proportionately to each agent based on their score within that contest. These tokens are activated if the agent is below their expected tokens for the community, but are given to the agent deactivated otherwise. Agents can choose to sell these deactivated tokens on the open market, or to keep them in hopes that they will be able to activate them in future contests.

A portion of newly minted tokens (chosen through community governance) go to agents through agent contests, and the rest goes to novices through participation in novice matches (explained below).

##### Economic Value of Tokens

Where Verity tokens get their economic value is an important consideration for the platform. Verity tokens having a non-zero price on the open market is important for a number of reasons:

* Allowing agents to sell reputation legitimately, disincentivizes agents from illegitimate methods of profiting on reputation that can destroy the predictive power of it, such as selling their account.
* Requiring agents to buy into similar communities isolates each community from attacks on adjacent communities, creating a cost for every subsequent community the agent would like to attack.
* The price of reputation creates an incentive for agents to participate in communities that may be socially but not economically lucrative such as non-profits, by giving them the chance to sell their tokens to people whom are looking to prove those same skills in for-profit ventures.

This means that it’s important to do some reasoning about the economic properties of reputation token, and where they may derive their value.

###### Pricing Activated Tokens

An activated token is any token which an agent purchases while still below their expected tokens for the community they’re purchasing from. The central reason that an agent would want to purchase activated tokens instead of earning them on their own is to be able to get more rewards faster from participating in contests.

Therefore, the price that an agent should be willing to pay for tokens is determined by the net present value of the rewards that an agent would get when buying the activated tokens directly minus the net present value of the rewards that an agent would get over that same period it would take them to earn those tokens through competing in contests. Thus the relevant factors for reputation price are:

* How long it takes to earn new reputation tokens
* How economically valued the reputation token is within contests.

This seems to match our intuitions about how reputation should be valued – Reputation that is harder to earn and more economically valuable tends to be seen as more valuable in society.

###### Pricing Deactivated Tokens

When looking to price deactivated tokens, similar considerations are taken into account. However, there are two additional caveats. Firstly, the amount of time that can be shaved off by buying deactivated tokens is significantly reduced because getting the benefit from them requires participating in contests. Secondly, there’s a third factor besides the hardness and economic values of contests – there’s the agents probability estimate of how many reputation tokens they’ll earn and lose in that period.

For the purposes of clarity, make three simplifying assumptions. Firstly, it will take agents twice as much time to earn tokens the normal way as it will to earn tokens by buying them deactivated. Secondly, this means the agent will be able to earn double the rewards over the same time period that it would take them to earn those tokens the normal way. Thirdly, their ability to earn those tokens is a simple binary situation where the agent either gains all the deactivated tokens with some probability, or loses all the deactivated tokens.

In this case, the agent should be willing to pay double the net present value of earning the tokens the normal way, multiplied by their own confidence interval that they will indeed earn all those tokens. Once again, this matches with our intuition of how reputation should be bought and sold in the real world, such as in the case of an agent buying a brand in the assumption that they’ll be able to maintain the same value as the previous owner.

###### Pricing Verity Clear Tokens

The price which an agent is willing to pay for Verity Clear tokens should track the price of the highest price activated token at any given time, as they can be turned into these tokens at no cost. However, this assumes that the value of buying token in an existing community will always be more than the value of creating a new community.

Any new website that wishes to use our technology in an entirely new community (and therefore get most of the early stake within that community) will want to purchase Verity Clear in order to seed that community with some agents. Thus the value that a community creator would be willing to pay is determined by the net present value of the expected future profits from that community (which they can get either by being an agent in that community, or a client.)

Thus whether or not Verity Clear tokens are being bought for community creation or reputation will fluctuate. As new use cases for Verity are created, it will be more attractive to create new communities to take advantage of those use cases. As the websites behind those communities begin to grow, it will be more attractive to transform the tokens into reputation tokens. Then, as the communities mature, this will in turn enable new use cases, which will again create the incentive to create new communities.

#### Onboarding Agents in Verity

Verity has two distinct ways to earn it’s tokens.

1. **Novice Matches**, involving contestants who are working to earn newly minted Verity colored tokens.
2. **Contests**, involving agents gaining and losing tokens from each other, in addition to earning newly minted and possibly deactivated tokens.

By separating the two ways to earn tokens, we have a smooth path for which agents can work towards earning money on the platform while learning community norms and improving their skills.

##### Novice Matches

Novice Matches allow new agents to grow their affinity score and earn Verity tokens as a new member of a community. They’re scored very differently from agent contests, and must use a separate Sybil defense mechanism because they can’t use token ownership as a proxy. Every contest contract can specify a corresponding novice match protocol, and if so, will have a mirrored version of all agent contests using that protocol. The general format of novice matches is as follows

1. Only accounts without Verity Tokens in that community can participate in the matches
2. All accounts must complete a nominal task such as a captcha to participate in matches for that month.
3. Only accounts with an affinity score of .5 can earn token in those matches.
4. Novice Matches are played out over a defined period. At the end of the period, all qualified participants are given a percentage of the newly minted Verity tokens, based on their performance in the novice matches that month.

#### Skill Types and Contest Rulesets

Any community can choose a skill tag that it chooses to measure using its reputation tokens, provided it has some Verity Clear tokens that it can seed its members with. The community simply chooses what **contest ruleset** it would like to allow to for its skill tag, and a variety of different behaviors emerge that can measure very different **skill types.** To start with, Crystal will provide three sets of rules around contests, which allow three very different options for what skills mean. These three different rulesets allow a very flexible definition of skills that can encompass most use cases.

##### Guild Skills

A **guild skill** in Verity allows any contest that has consistent values, but a customizable utility function. In practice, what this means is that the community defines what type of expertise it would like to represent, and then it allows any client to have its members fulfill that client’s goals. The guild’s challenge in this case is to define their values in such a way that they believe strikes a good balance in flexibility of clients vs. specialization of skills.

Figure - Style Attributes in 99designs

A real life example of the guild skill model would be 99designs.com. 99designs handpicks what it calls “Platinum Designers” – designers whom meet the criteria that 99designs values in designers (99designs, 2016). At the same time, 99designs allows for clients to choose their own criteria for their utlity function, using what it calls “style attributes” to describe what features the client is looking for.

Instead of handpicking designers one by one, with Verity one could imagine a model where 99designs handpicked a few people it considered “good designers,” and use this as the start-set for the “good design” value. New platinum designers could then grow organically as members within the community rated each-other with that good design value.

##### Shared Skills

A **shared skill** is a flexible skill type in which there’s some broad agreement around what goals a skill is trying to accomplish, but there many different values that different communities have that further refine that skill for their particular community. In practice, what this means is that the community defines a standard utility function for every contest, but allows each contest to specify its measurement for affinity, by which the skill tokens get weighted for that contest.

While there are currently no real life examples of shared skills, this is simply because no system like Verity currently exists. An example of where a shared skill might be useful is for a skill tag called “internet commenting.” A simple utility function could be defined, which would be a consensus contest where the goal was to up-vote comments that other members also up-voted. This same skill could be shared by Huffington post, The New York Times, Reddit, and YouTube.

In order to capture the different types of comments that might be considered valuable on different platforms, an agent’s tokens can then be weighted by affinity to each community’s own values. For instance, humor might be valued very highly in Reddit, but lower in The New York Times. Alternatively, YouTube and Reddit might both value humor, but have different ideas of what constitutes good humor. This mechanism captures these differences quite elegantly.

##### Organizational Skills

**Organizational skills** are skills in which both the goal and the values are fixed. In practice this means that an organization has its own utility function, and people can prove their value to that organization by having values the organization values, and helping it maximize its utility. The tag will be something like ‘Adding Value to Verity’, and the amount of tokens an agent holds would represent how valuable they are to the Verity organization.

An example of how this works in the real world is pay scales. An organization tries to find people that fit in with its culture, then pays them based on how much value they add to the organization.

#### Contest Collaboration

While contests are a powerful tool, they suffer from a key deficiency – they encourage competition, but as a result discourage collaboration. To solve this, Verity introduces the notion of **shared credit**, and a particular type of shared credit called sub-contests. These two mechanisms together allow for flexible collaboration in which people share in the rewards for a particular submission, critique, or prediction.

##### Shared Credit

Verity's shared credit mechanism allows a smart contract to act as a proxy for any number of agents, whom each can contribute some of their skill tokens to the contract, without deactivating or otherwise jumping through the hoops that are normally required to transfer tokens. In the simplest example, a team of three people could work together to create a submission, prediction, or critique. They could use a shared credit smart contract that took the rewards due to that submission, and split them evenly three ways when they eventually decided to split their team up.

###### Sub-Contests

Sub-contests are a particular type of shared credit rule that allow people to work off of other's creations. This allows the shared credit mechanism to represent situations like Wikipedia and git, where people are taking the best product and working to improve It, rather than competing to create the best product from scratch. The general idea behind subcontests is that each submissions itself becomes a contest, in which derivatives of that submission are competing to be the best “version” of that submission. By averaging the percentage improvement in utility of each solution with the percentage similarity with each submission, we can figure out how much of the credit should go to the original solution, and how much should go to each of the sub-solutions.

By combining this mechanism with the simple shared credit mechanism, very complex behavior can be created, including behavior in which several solutions from sub-contests are combined into a single maximal solution, and skill tokens are distributed accordingly.

For example, imagine a group of editors collaboratively editing a chapter of a book. The chapter would be a submission, and a few sub-contests might emerge to edit various paragraphs into more readable versions. The best paragraphs from each content could then be combined, allowing for the best chapter overall to emerge.

## Contribution-Based Reputation

Contribution-based reputation is the third core type of reputation. While organizational skills-based reputation measures an agent’s skill at adding value to an organization, contribution-based reputation actually measures how much value that person has added – that is, it takes into account the amount of work actually done. By using Verity contests to create clear measures for each action, a transparent, accurate method of ranking contributions emerges.

### Utility Function Standard

In order to issue contribution-based reputation, a community must first choose a utility function.

A utility function is a function written in solidity that outputs a number. It indicates the community’s preferences and goals – the higher the number, the more the community wants that outcome. Contribution tokens are handed out based on how much utility a particular agent has contributed to the community.

A community’s utility function is chosen according to the governance mechanisms in that community, and can evolve over time as the communities’ goals and preferences change, or the community realizes that their initial utility function didn’t fully specify those goals and preferences.

All utility functions follow Verity’s utility function standard – a standard framework that can be used by any governance mechanism that requires mathematical precision around goals (such as futarchy).

The utility function standard gives a serialized format that explains all input variables, a simple function to plug all relevant variables in that returns the utility, as well as a standard functions to deal with specialized utility (explained below).

#### Specialized Utility

One common thing that communities want to do is measure specialized tasks, actions or roles. While a general utility function is needed at the community level, it may make sense to create customized utility functions that relate a community’s goals to specific metrics or actions in an agent role. For instance, it may make sense to measure a phone support rep by customer satisfaction and calls per hour, and then have a standard way to relate these metrics to the broader community goals.

To handle this, Verity allows one to create specialized utility functions that relate to a specific task, and then plug those functions as inputs directly into the main community utility function. This process can be nested such that a specialized utility function plugs into another specialized utility function, and so on. This can be used to create agent metrics for specific roles, to give standard rewards for things like referring new members to the community, or even give a standard one-time reward for doing something like reading the employee manual.

### Contribution Tokens

Contribution tokens are used to measure how well an agent helps a community meet its utility function. Contribution tokens work by assigning a value to actions that community members take, as measured by the utility function. Each community may issue its own contribution tokens, and allow contribution tokens to be transferred or forbid transfer and lock tokens into the agents who earn them.

#### *Calculating Contribution Tokens*

Contribution tokens are calculated by using Verity contests to measure the utility of a contribution. In the absence of contests, specialized utility can be used to give contribution tokens for certain actions as well. When a contest is complete, contribution tokens are minted and issued in accordance with how much utility a specific contribution gives to the community. Optionally some amount of contribution tokens can also be issued for accuracy, and critiques (using specialized utility) which are then distributed according to the rules of the contest.

# Governance Layer

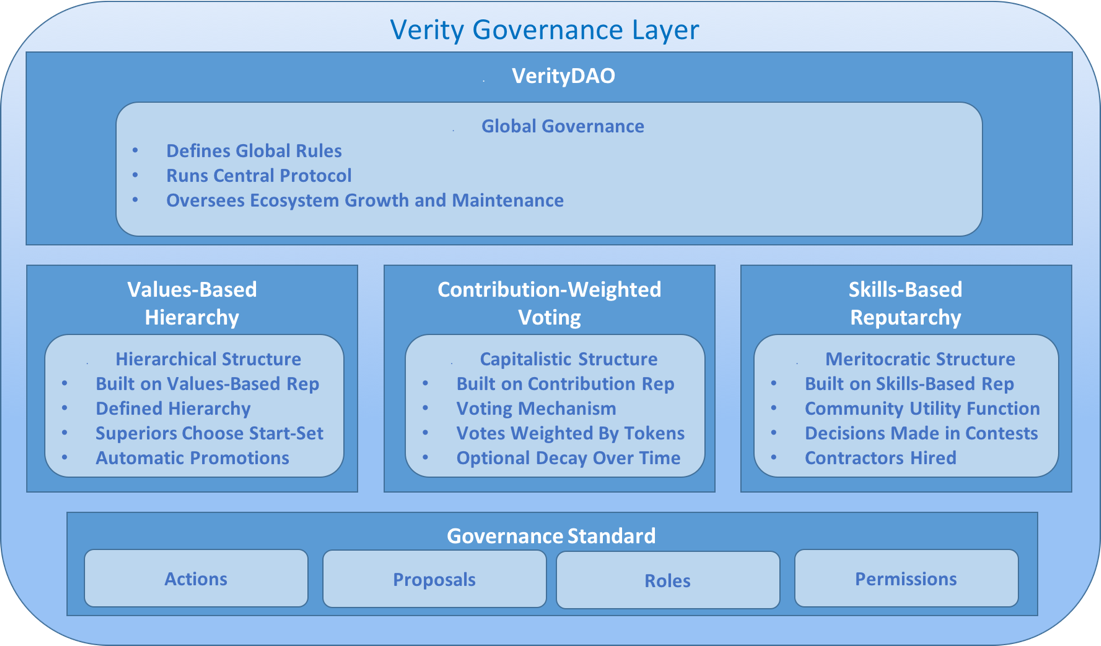


Figure - Governance Layer

Verity’s base governance layer at its core is a simple standard that can be extended with smart contracts to encompass arbitrary governance protocols. This standard defines governance for each agent Verity Community, as well as governance at the protocol level, called VerityDAO, which defines global rules about interactions between and within every community

On top of this standard we build three primary forms of governance. The first is a hierarchical structure based on values-based reputation. The second is a capitalistic structure based on contribution-based reputation,. The third is a meritocratic structure using skills-based reputation.

For agent communities, governance is decided at the point of creation. The plan for the VerityDAO is to start with a hierarchical structure, in which the decisions are informed by skills-based reputarchy, and build in incentives (similar in spirit to Ethereum’s difficulty bomb[[2]](#footnote-3)) to switch to voting with skills-based reputarchy and paying people with contribution-weighted payouts as the platform matures.

## Governance Standard

Verity’s governance standard is a simple flexible standard that includes hierarchical roles, permissions, actions, and proposals. The systems also allow roles to be held by smart contract agents, which themselves can follow the governance standard. By layering these simple building blocks together, complex governance mechanisms can be built, and mechanisms like Futarchy, Liquid Democracy, or Quadratic Voting can be easily swapped in and out as desired. Nexusdev’s dappsys framework[[3]](#footnote-4) already includes much of this functionality, and we’re currently investigating the possibility of adapting it for our purposes.

### Actions

An **action** as defined by Verity is a specific function that lives within a smart contract and can be executed on the Ethereum network. The action could be a simple action that merely triggers an event that shows that the action happened, or can be a complicated action that involves invoking outside contracts or moving ether. The action may take arguments, which are inputs to the action that change its meaning, or they may simply be actions that have one meaning, such as “resign.”

An example of an action in the US government might be “approve citizenship” which takes as an argument the identity of a potential citizen.

### Proposals

A proposal in the governance standard is a suggested action (with suggested arguments) that has not yet been executed. In this sense, the action is being proposed, and the governance standard can define any number of potential paths and states through which it can go through to ultimately be rejected, or accepted and executed.

#### Proposal States

A proposal state represents a proposal’s status within the governance process. Every action has two states by default: ‘rejected’ and ‘approved’. A rejected proposal is a proposal which is dead, and can longer change states. An ‘approved’ proposal is a proposal which will be executed on the blockchain. In Verity’s governance standard, any number of arbitrary states may be created representing arbitrarily complex governance rules.

For instance, in the US government, when the action proposed is “create bill” (and the bill itself is the argument), the proposal would go through the “senate vote” state, the “house vote” state, and the “presidential veto period” state, before finally being accepted or rejected.

### Roles

A **role** in the governance standard is a specific function that can be assigned to any agent in the system. For instance, roles in the US government might include Citizen, President, Senator, and Congressman. Other non-obvious roles might include The Constitution, The Senate, and the House; these roles would be filled by smart contracts, which themselves could implement their own governance rules.

Verity’s standard also supports hierarchical roles. A child role will inherit all permissions from its parent role unless otherwise specified, as well as any additional permissions assigned to it. For instance a CEO can inherit at the very least all the permissions that a manager has, and a manager can inherit all the permissions of an employee.

### Permissions

**Permissions** in Verity are given to specific roles and give that role the ability to participate in certain ways within the governance system. There are only two types of permissions within the governance system; roles can be given the permission to execute an action, and they can also be given the permission to change a proposal from one specific state to another specific state.

In the US government, a “senator” role might have permission to execute a “vote in the senate” action, and a “the senate” role might have permission to change a “create bill” proposal from the “senate vote” state to the “house vote state” or the “rejected” state.

## Governance Systems

While Verity can represent any governance system (that can be represented in a Turing complete language), it will come with several options out of the box, which make unique use of Verity’s own reputation metrics. These reputation-based governance methods represent new paradigms that can work equally well in traditional settings, as well as decentralized applications.

### Value-Based Hierarchy

In a **value-based hierarchy**, every role in an organization is matched with a value tag, such as “good customer support person,” “good customer support manager” or “good engineer.” The roles exist in a hierarchy, such that executives exist above customer support managers, and customer support managers exist above customer support employees.

The central idea behind value-based hierarchy is that each stage in the hierarchy gets to choose the start-set for the role below it. Thus executives choose exemplary examples of managers, managers choose exemplary examples of employees, and so on. When someone’s value score rises above zero for a given role, they are automatically given all the permissions and perks associated with that role.

This allows the entire organization to participate in the promotion process and be on the lookout for ideal employees, while still giving higher-ups the ability to choose what type of qualities they’re looking for in each role, and prizing the opinions of agents already in the role.

### Contribution-Based Voting

In **contribution-based voting**, agents vote on actions, and their votes are weighted by the amount of contribution tokens that agent holds within that community. This can be seen as a meritocratic system in which those whom offer more value into the community get more say within it. This voting method can optionally have contribution tokens’ weight decay logarithmically over time, such that people whom have given more recent contributions are given more weight than those whom gave contributions long ago in the organizations history.

### Skill-Based Reputarchy

**Reputarchy** uses Verity’s reputation and community mechanisms to give decisions to those whom have proven they’re best at making them. If futarchy is described “vote on values, bet on beliefs,” (Hanson, 2003) then reputarchy could be described as “vote on values, crowdsource beliefs.” The community decides on a global utility function, and then decisions are made by creating Verity contests that maximize that utility function. Optionally, contractors can apply to enact those decisions, and their proposals will be ranked as well using Verity contests.

In reputarchy, every community chooses a utility function. Then contests are created which measure various decisions related to that

#### Reputarchy in Action

Imagine an “end global warming” community that made a utility function with two variables: how much carbon emissions are lowered within five years, and how ethical each action is considered by ethical members of the community. It would then choose guilds and shared skills which it thought it might need, such as choosing the “Washington lobbyists” guild skill for the “lobbying” skill tag and the “high-tech inventors” shared skill for the “technology development” skill tag. The community could then create a contest for the best lobbying strategy that met its utility function of lowering carbon emissions, and have the inventors work together to come up with potential promising approaches to reducing carbon emissions. It would choose those actions which had the highest overall reduction in carbon emissions, and hire contractors to execute on those plans.

## VerityDao

The **VerityDAO** is the organization which is in charge of running the central protocol and overseeing the growth and maintenance of the entire Verity ecosystem. VerityDAO plans to start with a simple multisig governance mechanism, informed by Reputarchy style suggestions, then later switch to fully Reputarchy-controlled governance mechanism with contribution-based rewards as the ecosystem of agent communities matures.

### Making a Profit

VerityDAO makes money by taking a small cut of the reward from every paid contest in every community. This reward then goes into the Verity smart contract, where it can be used to help fund new communities, bolster existing communities, create better infrastructure, or be passed on to agents in the form of weekly payments.

### VerityDAO Governance Actions

As the parent organization, VerityDAO is in charge of taking actions that cause the ecosystem to flourish. What follows is a list of some of the most important actions that VerityDAO can take

#### Setting Verity Token Issuance Rate

Issuance of new tokens in Verity is set at a steady monthly limit, which can be tweaked by the stakeholders. This is analogous to a central bank which is incentivized to maximize the value of the tokens. Newly issued tokens are split percentage wise among all the communities, and are created to fuel novice matches and allow the onboarding of new agents into Verity communities.

#### Setting Verity Token Buy and Burn Rate

Just as it may be prudent to issue more tokens in order to onboard more agents onto the platform, it may also make sense to shrink the supply in order to raise the value of Verity tokens. For this purpose, the Verity has an ongoing offer to buy and burn tokens at a price set by the DAO. While this price will typically be set to 0, it may be raised above market value if the supply is growing too fast, in order to incentivize burning of tokens and shrink the supply.

#### Choosing Core Values

The VerityDAO is in charge of choosing its own core values. These core values will in turn determine the value-weight of all votes in the parent organization, based on those voters’ affinity scores with the organization's core values.

#### Adding and Removing Member From the Start Set

For each of the core values, the VerityDAO can choose members whom they view as paragons of those values, and add them to the start set. If it comes to light that a person has a different character than previously assumed, they can be removed from the start set.

#### Setting Contest Fee

Every contest on the Verity platform has a small percentage fee taken from it, which goes to the parent organization. The VerityDAO sets this fee at such a level to be competitive, while still allowing maintenance and growth of the platform.

#### Setting Contribution-Weighted Payment

Every week, a small amount of earnings are taken from the VerityDAO and issued to stakeholders, weighted by their contribution tokens. The VerityDAO votes on this amount.

#### Updating the Protocol and Governance Rules

The VerityDAO is in charge of creating updates to the protocol, as well as to its own governance rules.

#### Sending Money

The VerityDAO can send money to any account on the Ethereum platform. This may be in payment to a contractor, as a stimulus to an agent community, or for any other reason.

# Collaboration Layer

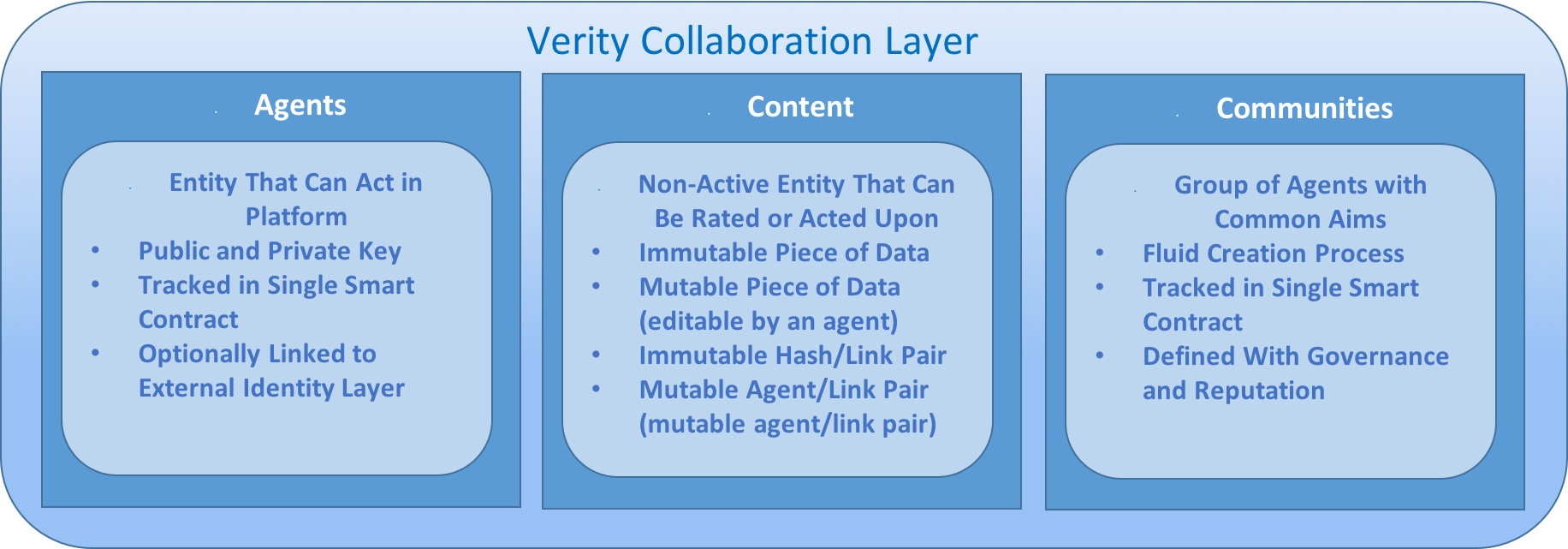


Figure - Collaboration Layer

Taking a broad overview, the Verity Platform provides the underlying Governance and Reputation infrastructure to build, operate and scale advanced highly collaborative communities, made up of diverse agents. The **collaboration layer** of Verity involves systems to create, read, update and delete agents, content and communities within the system.

## Agents

An **agent** in Verity is any entity that can perform actions in the system. Agents can be assigned roles and take actions within a given governance system, and receive scores and rate other agents within a given reputation system; both the reputation and roles of an agent are specific to the communities which they are a part of.

Agents in Verity consist of a public key tracked entity in a single smart contract. The Verity protocol makes no assumptions about who controls the private key, nor any assumptions that tie the public key to real world identities. Instead, Verity allows any identity and key management system to layer on top of its governance and reputation protocols, allowing for a variety of identity solutions depending on the use case of a particular community, governance system, or reputation protocol.

The Verity protocol is specifically engineered such that any part of the system that can take actions (including communities and VerityDAO) are automatically registered in the agent registry. This ensures interoperability between communities, and allows for future integrations where for instance communities themselves can create meta-governing bodies and receive reputation.

## Content

**Content** in Verity refers to any entity in the Verity ecosystem which can be rated or acted upon, but which itself can't take actions. By creating a central repository for content, one can link an individual's view on a piece of content across communities, and aggregate the opinions of various communities of experts on a single idea or action. Semantic information about the content can be created using the attestation or reputation standards, or can be layered on top using other systems like IPFS's IPLD[[4]](#footnote-5). For instance, a single legal policy can be added to the content database as a piece of content, and using attestations, a tag cloud can be created that shows that it is a piece of legal content. Then using attestations and reputation, experts on climate change, foreign policy, and social justice can all separately weigh in from their respective communities on the impacts to their area of interest. Because of the central content registry, these opinions and forecasts can be aggregated into a single user interface. The same can be done with reviews and ratings of things like restaurants.

Content tracking in Verity is expressed as a single searchable smart contract that allows the registry of content. The content can be of four types:

1. An immutable piece of data stored directly in the smart contract.
2. A mutable piece of data editable by an agent and stored directly in the smart contract.
3. An immutable hash/link pair that links out to an off-chain piece of content and can be checked against the hash to make sure it has not changed.
4. A mutable agent/link pair that links out to a mutable piece of content signed by the owner agent to ensure its legitimacy.

The latter two types of content can link out to private sources, and the former two types of content can be encrypted, allowing for various levels of private content. Together, these four types of content can cover nearly all use cases. The Verity protocol is specifically engineered such that all non-active entities within the system (such as attestations and value tags) are automatically registered in the content database, allowing these items to also be rated and given reputation throughout the system.

## Communities

**Communities** are places where agents can gather, converse, and work together to provide value to each other or to clients. Every community has different goals and purposes, defined by the agents that are part of that community, and the core values of the community.

### Creating a Community

Communities are created in stages. First, the idea for a community is proposed. Potential participants in the community will begin to interact with each other, tagging each other with value tags as they discuss the community.

As the set of ideas formulate into actionable purpose and mission, a group of founding members (or a single founder) will initialize the community by choosing a governance mechanism, initial reputation mechanisms and applicable core values. Core values can be chosen by looking at commonly tagged values between founding members, or can be ignored and uniformly chosen by the founder(s).

After the initialization period, the community actively begins performing work toward their mission. To accomplish initial startup activity, community members begin creating and participating in contests. At this point, self-governance community rules, values and behavior reinforce and amplify the desired community practice toward their mission. As needs arise, community practice can evolve or be tailored according to chosen or amended community governance protocols.

As ongoing practice progresses, community members further interact in contests and campaigns performing work toward their mission which reinforce and amplify metrics in **values-based reputation**, **skills-based reputation** and **contribution-based reputation**. New Verity tokens can enter the community by transforming according to the normal token transformation rules explained above.

### Community Governance Actions

#### Choosing a Utility Function

The community is in charge of choosing its own utility function. This will be used when creating contests, and also for tracking contributions of agents to the community.

#### Choosing Core Values

The community is in charge of choosing its own core values. These core values will in turn determine the value-weight of all votes in the parent organization, based on those voters’ affinity scores with the organization's core values.

#### Adding and Removing Members From the Start Set

For each of the core values, the community can choose members whom they view as paragons of those values, and add them to the start set. If it comes to light that a person has a different character than previously assumed, they can be removed from the start set.

#### Setting Contest Fee

Every contest in the community has a small percentage fee taken from it, which goes to the community. The community sets this fee at such a level to be competitive, while still allowing maintenance and growth of the community.

#### Choosing Contest Types and Skills

Every community can choose skill tags it would like to track, and choose whether they would like to use existing measures or create their own. If they choose to measure the skills themselves, the community must also choose the contest types it will allow to use those skills.

#### Updating Contest and Governance Rules

The community can choose to update its privileged contests to new versions, as well as changing its governance contract.

#### Sending Money

The community can send money it owns to any account on the Ethereum platform. This may be in payment to a contractor, or for any other reason.

# Applications Layer

The Verity **applications layer** refers to the ecosystem of mobile apps, decentralized apps, and web apps that implement the Verity protocol. Together, these applications form a web of communities, focused around a variety of aims, that can interact with each other through common governance protocols and create shared definitions of reputation.

The Verity protocol aims to be application and business model agnostic, working equally well with mobile, desktop, and web protocols (as well as future platforms like VR and AR). Verity communities can be global or localized, public or private, centralized or decentralized. By building on the Verity protocol, these diverse applications can still interact with each-other and benefit from ecosystem innovations. Developers may onboard subsets or whole layers of the Verity protocol within their existing projects or A/B market testing activities.

Fundamental properties built-into the Verity ecosystem are a) credible reputation metrics, b) flexible governance settings and c) all agent, content and community specific central repository data is internally self-generated and maintained within the Verity ecosystem. Within the Verity protocol, the responsibility of maintaining proper incentives and managing “sensitive” user data is offloaded to the protocol itself, turning everyday behavior into verifiable metrics, while allowing applications to focus on core functions related to their purpose. The benefit is happy users aligned with clearly defined and accepted governance, faster time to market (for communities and applications), reduced burden of community management (applications no longer need to build and maintain this on their own), while expanding access to interconnected tools, processes and networks.

## Example Applications

### Decentralized Applications

Decentralized applications (Dapps) have a particular problem that other applications don’t have. They often need to make decisions that require human judgement, but giving the decision to any one person or small group of people can end up re-centralizing that app by concentrating power. With Verity, all of these human judgements can be made in decentralized way. For instance, one can **moderate content** without a centralized moderator, **settle agent disputes** without a single company to handle support, **identify trusted agents** for distributed work, and even create **truth oracles** that provide trusted data without relying on a single source.

### Rating and Grading

Ratings and grades are notorious for creating adverse incentives that cause bad behavior. The centralized review site Yelp has been accused of extorting business to bury negative reviews[[5]](#footnote-6), and teachers have been known to give their students the correct answers on standardized tests[[6]](#footnote-7). With Verity, you can **democratize education** through decentralized assessments, create **cheaper insurance** through crowdsourced risk assessments, and **disintermediate review websites** with reviews and ratings that can't be gamed.

### Collaborative Crowdsourcing

Anything that can be created by an agent on a computer, can be created by Verity crowdsourcing. With Verity, the things built are optimized to meet your exact goals or preferences, and are typically better results than you would receive by contracting a single agent. You can **have software reviewed by thousands of reviewers.** You cancreate a **compelling creative book**, and give royalties based on how much each person contributes. You can even **design** **cars**, **buildings** or **organizations**, all at a cheaper price point and likely greater quality than an individual agent.

### Portable Reputation

With Verity, you can earn reputation once, and take it everywhere. This means that **low quality internet comments will be buried,** as the best commenters earn their reputation and have the means to consistently present their reputation from site to site. Instead of needing to prove yourself on each new **marketplace,** you can take it with you (for example transfer reputation from Amazon to eBay). Finally, the **sharing economy** will benefit from higher credential confidence regarding safety and quality of work protections. Broader visibility into Uber, Airbnb, and Snapgoods reputation will all be linked, allowing every agent to get consistent, reliable and high quality service they expect.

### Decision Making

With Verity, making decisions will be decision-theoretically optimized according to your goals. Want to get **agent advice** on buying shoes that look great and last forever? With Verity, the crowd will find the perfect pair for you that balances those two goals. **Business strategy** and market needs analysis will forever be changed as companies find out that their CEOs’ decisions can be matched or beaten by communities, with lower price points. Eventually, **countries** may even jump on the reputarchy bandwagon (some noteworthy parties and elections may need improvement to gain alignment with their respective constituents).

### Forecasting

Knowing future possibilities is crucial to wellbeing of millions of people. Wouldn’t it be great to know the likelihood of a major war springing up in the next year or the chance of an earthquake in California in the next six months? With Verity, **large events, natural disasters,** and **existential risks,** can all be given precise probabilities. These probabilities have been shown to outcompete even the top forecasting agents, and are of similar quality to prediction markets.

### Machine Learning Augmentation

Training machine learning systems to interact with human systems is a common goal, but the existing approaches suffer from limited training data and poor specification of human values. Verity can solve these problems and create a synergistic effect between machines and humans. Using the Verity protocol, for the first time there’s an **API for human ingenuity**, which communities can be plugged into in order to solve problems machine learning algorithms can’t. The machine learning algorithms can then use these **contests as training data**, significantly reducing the effort needed to compile such a data set. Finally, these machine learning algorithms can be used to further assist humans, completing the cycle. In general, it is assumed that participants in Verity communities can be human, AI or humans augmented by AI.

# Conclusion

For the first time, Verity makes possible self-governed generation of credible and immutable reputation metrics that represent attributes and work we care about in the real world. Our value rank algorithm allows any humans’ values, skills and contribution to be quantified using the notion of transitive trust. Our Verity tokens allow any humans’ skills and contribution to be quantified using basic decision-theoretic criteria. By combining this reputation with a novel governance method for flexible crowdsourced contests, we can harness the wisdom of the crowd to out compete agents. This has far reaching implications for governance, decision making, forecasting, crowdsourcing and society at large.

The Verity platform aims to fundamentally redefine trust and collaboration on the internet, allowing users and applications to work together towards shared values and goals, and share in the profit the platform earns. Verity enables social applications which depend on verifiable, trustworthy and collaborative communities. Applications benefit from happy users aligned with clearly defined and accepted governance, faster time to market, reduced burden of community management and expanded access to interconnected tools, processes and networks.

# Bibliography

99designs. (2016, October 28). *How to become a Platinum designer on 99designs.* Retrieved from 99designs Blog: https://99designs.com/blog/inside-99designs/platinum-designer-99designs/

*Aggregative Contingent Estimation*. (2016, September 30). Retrieved from IARPA: https://www.iarpa.gov/index.php/research-programs/ace

Atanasov, P. D., Rescober, P., Stone, E., Swift, S. A., Servan-Schreiber, E., Tetlock, P. E., et al. (2015, October 8). Distilling the Wisdom of Crowds: Prediction Markets versus Prediction Polls. *Management Science* .

Baron, J., Mellers, B. A., Tetlock, P. E., Stone, E., & Ungar, L. H. (2014). Two reasons to make aggregated probability forecasts more extreme. *Decision Analysis* *, 11* (2), 133-145.

Buterin, V. (2014, March 28). *SchellingCoin: A Minimal-Trust Universal Data Feed.* Retrieved October 17, 2016, from Ethereum Blog: https://blog.ethereum.org/2014/03/28/schellingcoin-a-minimal-trust-universal-data-feed/

Culbertson, S. S., Henning, J. B., & Payne, S. C. (2013). Performance Appraisal Satisfaction: The Role of Feedback and Goal Orientation. *Journal of Personal Psychology* , 189-195.

Fan, X., Liu, L., & Li, M. (2012). EigenTrust++: Attack resilient trust management. *2012 8th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)* (pp. 416-425). Curran Associates.

Galton, F. (1907). Vox Populi. *Nature* , 450-451.

*Google Scholar Search*. (2016, October 30). Retrieved from Google Scholar: https://scholar.google.com/scholar?ion=1&espv=2&bav=on.2,or.r\_cp.&biw=1366&bih=667&dpr=1&um=1&ie=UTF-8&lr&cites=4085997296011241419

Gun Sirer, E. (2016, October 17). *Micropayments an Cognitive Costs.* Retrieved from Hacking, Distributed: http://hackingdistributed.com/2014/12/31/costs-of-micropayments/

Hanson, R. (2012). Logarithmic markets coring rules for modular combinatorial information aggregation. *The Journal of Prediction Markets* *, 1* (1), 3-15.

Hanson, R. (2003). Shall We Vote on Values, But Bet on Beliefs? *Journal of Political Philosophy* .

Jose, V. R., Nau, R. F., & Winkler, R. L. (2008). Scoring rules, generalized entropy, and utility maximization. *Operations Research* *, 56* (5), 1146-1157.

Kamvar, S. D., Schlosser, M. T., & Garcia-Molin, H. (2003). The eigentrust algorithm for reputation management in p2p networks. *Proceedings of the 12th international conference on World Wide Web.* ACM.

Nesta. (2016, November 11). *The History | Longitude Prize.* Retrieved November 11, 2016, from Longitude Prize: https://longitudeprize.org/history

Resnick, P., Zeckhauser, R. J., Swanson, J., & Lockwood, K. (2006). The Value Of Reputation On Ebay: A Controlled Experiment. *Experimental Economics* , 79.

Spiegel, A. (2016, September 30). *So You Think You're Smarter Than A CIA Agent*. Retrieved from NPR: http://www.npr.org/sections/parallels/2014/04/02/297839429/-so-you-think-youre-smarter-than-a-cia-agent

Szabo, N. (1999). Micropayments and Mental Transaction Costs. *2nd Berlin Internet Economics Workshop.*

Sztorc, P. (2016, September 12). *Oracles are the Real Smart Contracts.* Retrieved from Truthcoin Bog: http://www.truthcoin.info/blog/contracts-oracles-sidechains/

Traupman, J. (2007). *Robust Reputations in Peer-To-Peer Markets.* University of California, Berkeley, Computer Science Division.

Vogelsteller, F. (2016, September 30). *ERC: Token standard*. Retrieved from Github: https://github.com/ethereum/EIPs/issues/20

# Appendix A – Mathematical Building Blocks

## Utility Functions

A utility function is a mathematical representation of preferences. In decision theory, they're central to the concept of making sound decisions, and for that reason they're the basis of Creativity tokens in Verity. One's creativity will be scored based on how well one can generate options that maximize a given utility function. In addition, for each input into a utility function, a contest will be created to earn Accuracy and/or Creativity tokens.

Because there are near limitless representations of utility functions, Verity must allow for the possibility of arbitrary complexity. Given this constraint, each utility function will be its own smart contract on the Ethereum platform. These smart contracts will have standard functions, which input any number of variables through a contest format, and output a number. The higher the number, the more utility received.

Utility functions are hard to grasp for the average agents, and we’re actively looking into easy ways to intuitively create utility functions, including LENS modeling, willingness to pay models, visual equation builders, and premade utility function libraries. It’s also worth noting that for many applications, utility functions won’t have to be created by the agent, as the utility function will be premade by the front-end dapp, as in our examples with smart-contract security.

## Scoring Rules and Divergence Measures

A scoring rule is a rule which incentivizes participants to be accurate to create good forecasts by giving higher score to more accurate forecasts. A strictly proper scoring rule is any scoring rule in which incentivizes participants to be honest about their true probabilities, by giving the highest score to the most accurate predictions. In 2008, two families of strictly proper scoring rules were found to be able to incorporate decision-theoretic notions of risk-tolerance, information-theoretic notions of information gain, and approximate all of the popular scoring rules(Jose, Nau, & Winkler, 2008). These constructions were extended to incorporate the property known as sensitivity-to-distance which allowed forecasters whom were closer to the correct answer to receive higher scores, as well baseline distributions, which allowed for the notion of prior information.

The incorporation of a baseline distribution also allows the possibility for emulation of market scoring rules, a creation of Robin Hanson (Hanson, Logarithmic markets coring rules for modular combinatorial information aggregation, 2012) which use previous information gained from forecasters as the baseline.

By incorporating both the discrete and continuous cases of these two constructions, we can allow every contest to use the scoring rule that bests suits their needs, whether through a website choosing this option for their agents, or by creating a wizard through which a client can choose the scoring rule that best fits their needs. These constructions also map to a family of divergence measures that can be used for consensus contests.

## Pooling and Extremizing Algorithm

The pooling algorithm is the algorithm which combines probability distributions into a single distribution. A pooling algorithm can be as simple as a weighted average, known as linear pooling, with Accuracy tokens used for the weight. There are several other algorithms which are shown to be more accurate by engaging in a process known as extremizing, a process of moving aggregated distributions towards the extremes that has sound theoretical backing (Baron, Mellers, Tetlock, Stone, & Ungar, 2014). An active area of our research is around the tradeoffs to be made in terms of simplicity, gas costs, and accuracy in different pooling and extremizing algorithms.

## Monte-Carlo Simulation

A Monte-Carlo simulation for our purposes is a way to brute-force input the results from an aggregated probability distribution into a utility function. It can then be used to show how different submissions rank in regards to that utility function. Monte-carlo methods are very computationally expensive, and difficult to run on the blockchain. For more on how we’ll deal with this, please see **Appendix C – Technical Details and Challenges**.

# Appendix B – Math Formulas and Algorithms

The following sections denote some of the main algorithms needed to implement the Verity protocol. To facilitate ease of reading, the following symbols will be used consistently represent different concepts in Verity.

***Definitions***

|  |  |
| --- | --- |
| : The set of all agents | **:** A specific agent |
| : The set of all communities | : A specific community |
| : A specific value | : A specific skill |
| : Some amount of skill tokens | |

## B.1 – EigenTrust Algorithm

1. A **local trust value** is computed for every agent the originating agent interacts with. This is done by taking the number positive interactions with that agent minus the number of negative interactions.

***B.1 - Definitions***

satis(i,j): Satisfaction measure of transactions peer *i* had with peer *j*

sat(*i,j):* Number of satisfactory transactions peer *i* had with peer *j*

unsat(*i,j)*: Number of unsatisfactory transactions peer *I* had with peer *j*

*normloc(i,j)*: Normalized local trust value that peer *i* assigns to peer *j*

: The start-set of pretrusted agents in community c.

starttrust(i) = The start trust put into an individual peer *i* from the set P of trusted peers

*proptrust(i,k):* The propagated trust value that peer *i* assigns to peer *k*

1. A distribution is defined in which each member of the start-set is given equal trust.
2. The local trust score is normalized over all local trust scores of agents that this agent has rated, such that all the local trust scores sum to 1, with any trust scores below 0 being dropped entirely. If the normalized trust score is 0, the agent will spread it’s trust among the start set.  
   n
3. These normalized local trust scores are then aggregated, by assuming that trust is transitive. For a peer *k* that peer *i* has never met, peer *i* will add the normalized trust scores of all peers *j* whom have met peer *k*, and weight those scores by peer *i*’’s own trust in peer *j.*  
   proptrust()
4. This process is continued outward, such that peer k becomes the new peer j. Eventually, the agent has a complete view of the network. This can also be viewed as a probabilistic process, in which following agents with higher *t*’s will result in landing on a more trustworthy peer. This markov process can be modeled using linear algebra.

**Repeat**

|

|

|

**Until**

## B.2 – EigenTrust++ Algorithm

1. Average all local trust ratings together, between all agents *w* that agents *i* and *k* have both rated

***B.2 - Definitions***

trust(v,w): The amount of trust that node *v* places in node w.

*n*: The number of ratings that node v has made to peers *w*

*m:* The number of ratings that node u has made to peers *w*

sim(*i,k):*The feedback similarity that node *i* assigns to node *k*

*comn(i,k)*: The intersection of the peers that *i* and *k* have rated.

feed(i,k): The feedback score that peer i assigns to peer k

*fc(i,j):* The feedback credibility that peer *i* assigns to peer *j*

l(i,j): The normalized feedback credibility that peer *i assigns to peer j*

*normloc(i,j)*:As defined in B.1

edgeweight(i,j): Chance that peer *i*  will propagate its trust to peer *j*

1. Compute the similarity between two agents *i* and *k* by computing the sample standard deviations of all agent ratings that they have in common.
2. Divide similarity among all agents *m* that have a similarity score from agent *i* to create a normalized feedback score.
3. Define feedback credibility as a metric that weights each normalized trust score cij by each normalized feedback score feed(i,j).
4. Redefine local trust value lij in terms of its normalized counterpart that includes feedback credibility.
5. Redefine initial aggregated trust using the new definition of local trust given above.
6. Set the edgeweight for propagation such that it includes both similarity and normalized trust
7. Let Ri be a random number between 0 and 1 assigned to node i.
8. Activ
9. ∑ u∈AN(v) w(u, v) ≥ θ

## B.3 – Relative Rank Algorithm

***B.3 - Definitions***

A: The set of all peers that aren’t in the start set P.

P: As defined in B.1

*:* The subset of A where all peers have k feedbacks.

*r(k):* The top rated agent in the set

1. Separate start-set from non-start set members by taking the absolute complement of the start set.
2. Separate non-start set members into groups according to how many *k* feedbacks each member has received (both positive and negative)
3. Find the top rated agent in every group
4. Find a line of best fit for all pairs (k, rk), and determine the slope *m* and intercept *b* of that line
5. Define a non-start set agent *i*’s **relative rank** as an area relative to that line.

1. Repeat steps 3 - 5, but for set *P*.

## B.4 – Affinity Score Algorithm

1. Take the agents value rank along all core values for that community, and average them together.

***B.5 - Definitions***

affinity(): The function that calculates a user’s affinity score

*i*: The user who’s affinity score is being calculated.

*n*: The total number of core values within the community.

*c*: The community in which the affinity is being calculated

: A user *i’s* value rank in community *c*

## B.5 – Token Transfer Algorithm

1. Take the set of all agents who’s tokens *to* in community *b* are greater than 0

***B.5 - Definitions***

t: The function that calculates a user’s affinity score

: The user who’s affinity score is being calculated

*c*: The community in which the affinity is being calculated

*:* The subset of A where all peers have k feedbacks.

1. For each of these agents, create a set of pairs (*ta,tb*) for the tokens they hold in communities *a* and *b*
2. Calculate the Pearson Correlation Coefficient for all *n* pairs
3. For all *r* below 0, set the similarity score to 0.

Note that this similarity score is asymmetric because t > 0 for community *b,* but t ≥ 0 for community *a*. This has the effect of correcting the imbalanced similarity rate of large communities, which likely will have high many high similarity rates simply due to chance.

1. For all people that hold more than 0 tokens in community *j or k*, create the pair (*tj,tk)* where each *t* is the amount of tokens those people have
2. Use linear regression to calculate the line of best fit for all *n* pairs. Plug in the amount of tokens *t* to get the naïve rate.
3. Multiply the naive rate by the similarity score, which lowers the output of the linear regression when it applies less:

The expected tokens *e* are computed for all *n* communities the agent *u* belongs to with the following equation:

Where *s* equals the similarity score between community *j* and the desired community *c* and *t* equals the number of tokens in community *i*.

## B.6 - Token Transformation Algorithm

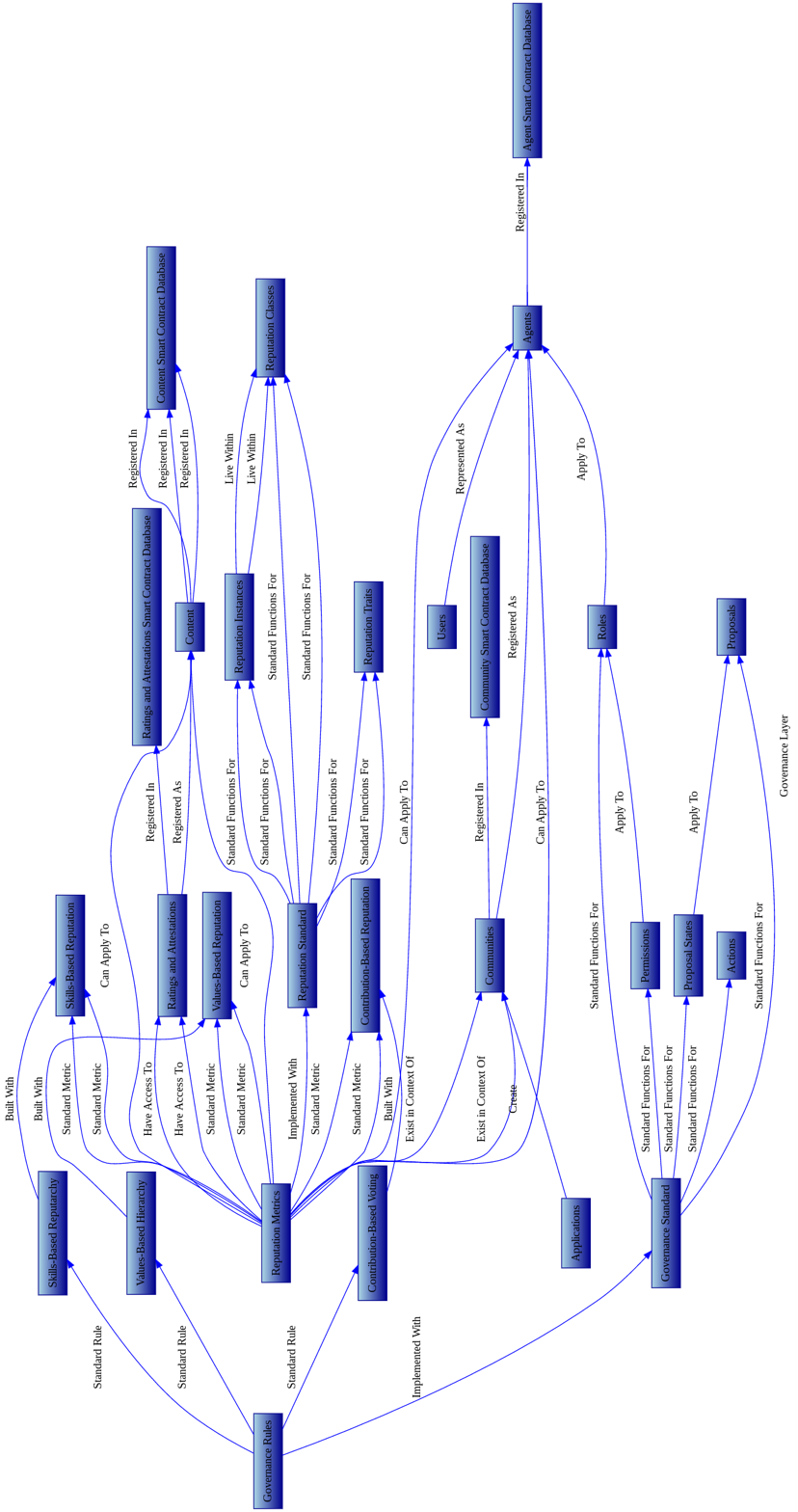


Figure - Verity Architecture

# Appendix C – Technical Details and Challenges

## Verity Architecture Flow Diagram

## Expensive Computations

Verity has a number of computations that would simply be too expensive to run on-chain. Particularly for Monte-carlo simulations, any decrease in cost and speed creates a corresponding increase in accuracy, so any optimizations made directly impact the quality of the platform. Because of this, it’s critical that the costs and speeds of computation are lowered from those typically seen on Ethereum.

On the other hand, the Ethereum blockchain provides availability, accuracy, and immutability guarantees that are critical for an application whose purpose is to quantify trust. It’s critical that these promises aren’t weakened too severely.

The paradigm that Verity plans to use to balance these two issues is to push expensive computations to off-chain, but to allow complete verification on-chain if there’s any disagreements.

This paradigm will be achieved through two emerging technologies: state channels and interactive verification.

### State Channels

The first technology is state channels. The idea behind state channels is that all parties involved in a transaction must agree on the results of a computation. If they all agree, only the results of the computation are posted on the Blockchain. If any party doesn’t agree on the results of the computation, it is then run on-chain to determine whom was correct. An early example of generalized state channels is the concept of “smart state channels” in Raiden.

#### State Channels in Action

One example of where state channels might be used is in determining the payouts for contest rewards and Verity tokens at the conclusion of a contest. If everyone agrees on the proper payouts within some specified time frame, payouts can happen cheaply and instantly, with only a small gas cost needed to update the final distribution of rewards on the blockchain.

If parties do not agree on the proper distribution of tokens, then the payout structure degrades to an on-chain method in which each participant is responsible for claiming their own rewards and therefore updating their own agent state within the smart contract. This method is more costly and time consuming, but it’s existence ensures that the reputation calculations can always be trusted.

The existence of Verity’s value-based reputation disincentivizes “griefing attacks” on the original state-channel based method, as agents whom submit wrong answers in order to raise costs for others will lose reputation, and therefore the ability to have financial stake in the community.

### Interactive Verification

Interactive verification works similarly to state-channels, but with a twist: Firstly, the computations need not be run by people involved in the transactions. Secondly, the full computation need not be done on chain, instead only enough needs to be run to determine which party is correct.

The way this works is that a computation is run completely off-chain by an interested party who posts up a security deposit. The results of this computation can then be challenged by another party, whom also posts up a security deposit. If challenged, the computation is then verified on-chain, and the incorrect party loses their deposit to the correct party. Early inroads to interactive verification were made with Ethereum computation market. More recently, TrueBit presented a method that works even for computations that would be too expensive for Ethereum’s gas limit.

#### Interactive Verification in Action

One place in which interactive verification would work for Verity is in its computation of Monte-carlo simulations. Agents (especially mobile agents) likely won’t have the computation power to run the simulations locally, and the full computation will most likely be above Ethereum’s gas limit. Both issues rule out state-channels as a viable solution, and thus require an interactive verification approach.

Taking TrueBit’s approach, this would involve submitting a Merkle tree of how one arrived at the final expected value of each contest submission (using a pseudorandom seed provided by the blockchain), as well as the expected value itself to the blockchain. A challenger whom thought the final computation was wrong would then be able to interactively challenge agent parts of the merkle tree where they thought the computation was wrong, and the blockchain would verify who was correct.

## Gas Costs

Gas costs on Ethereum are the costs associated with running computations and storing data on all the agents in the Ethereum network. While gas costs can be significantly reduced using the methods mentioned in the Expensive Computations section, they can never be completely eliminated.

This presents a challenge for a platform such as Verity that hopes to have broad appeal, as agents aren’t accustomed to paying for internet platforms, and even small costs have significant overhead in terms of the cognitive load (Szabo, 1999). Thus it remains important to enumerate whom will pay for gas, and how that payment will appear as choices for the agent. Verity has separate answers for these questions over the lifetime of the product.

### Short Term – Market Selection and Foundation Subsidies

In the short term Verity will take two approaches to ensure that gas costs don’t prevent adoption of the platform. Firstly, it will make a careful selection of its early markets, specifically targeting agents and developers whom understand the technology, and are willing to pay for the benefits of decentralization and transparency. In practice, this means targeting integration with other decentralized applications on Ethereum, whose agents and developers are both in this category.

Secondly, the Verity Foundation will work with a select group of partners to integrate with their existing crowdsourcing and sharing economy websites and apps, and completely subsidize the gas usage of these organizations for a number of years. Having immediate integration with the “legacy internet” and the existing “app ecosystem” is incredibly important for the future growth of Verity, and these integrations and subsidies will allow Verity to rapidly reach product-market fit within these key markets.

### Long-Term – Gas Automation and Website Subsidies

In the mid-term, as the reputation tokens begin to gain real-world value and contests with high rewards become common, Verity will work to create a system of smart-contracts that allows the agents to pay for their own gas without conscious interference. By automatically selling tiny fractions of the reputation tokens and rewards that a agent earns on a decentralized marketplace, Verity can create a system where agents are paying for gas costs without ever having to make the conscious decision to reach into their wallet. A system like this is necessary for widespread agent adoption in the long term.

In addition to agents paying their own way (which may not work for agents whom are performing poorly in contests), websites and apps can subsidize agents usage of their website by paying the gas costs themselves, in a similar way that they pay for computations now using a service like Amazon Web Services. At this point, Verity will have a proven use and track record, and can be thought of as a competitive advantage or even a cost-of-doing-business.

## Randomness

One issue with using probabilistic methods like eigenvectors and monte-carlo simulations to determine reputation is that a solid source of randomness is needed that can’t be manipulated by agents to gain more reputation. Non-manipulable randomness on the blockchain is hard to come by, and this represents a technical challenge that Verity must solve. To start off, Verity will use a simple semi-trusted oracle for offchain randomness such as OraclizeIt[[7]](#footnote-8). Over the long term, Verity would like to mix several sources of randomness together from many different actors with different incentives, such as OracleizeIt, Blockhashes, and a variety of RanDAOs[[8]](#footnote-9).

1. http://www.oraclize.it/ [↑](#footnote-ref-2)
2. http://ethereum.stackexchange.com/questions/3779/when-will-the-difficulty-bomb-make-mining-impossible [↑](#footnote-ref-3)
3. https://github.com/nexusdev/dappsys [↑](#footnote-ref-4)
4. https://github.com/ipld/specs/tree/master/ipld [↑](#footnote-ref-5)
5. http://www.huffingtonpost.com/news/yelp-extortion/ [↑](#footnote-ref-6)
6. http://abcnews.go.com/US/atlanta-cheating-178-teachers-administrators-changed-answers-increase/story?id=14013113 [↑](#footnote-ref-7)
7. http://www.oraclize.it/ [↑](#footnote-ref-8)
8. https://github.com/randao/randao [↑](#footnote-ref-9)