On Verifying Token-Based Systems

A Token Engineering perspective

Trent McConaghy · Following

Published in Ocean Protocol · 12 min read · Jul 14, 2020

> *"Trust, but verify" — Russian proverb*

Introduction

Decentralized Finance (DeFi) has exploded from a few hundred million dollars to billions under management in a few months. This is partly due to new incentives schemes in Compound, Balancer, and more. Consider yield farming. It's a loop of people pouring money, reaping near-term rewards, which they pour in as well.

Is yield farming safe? The answer depends on whom you ask. The underlying smart contracts have been audited, but those audits have nothing to say about *incentives*. The cases made for and against the safety of yield farming use simple logic and rhetoric. Yet there are literally billions at stake.

Are there better ways to verify incentives?

This isn't just a challenge for DeFi. Imagine that you've designed a Web3 ecosystem with incentives for long-term sustainability and growth. This is your livelihood — and your ecosystem's livelihood — at stake. How do you verify that design? In general, this is a challenge of *verification in Token Engineering (TE)*.

Defining the Problem

***TE Verification* is about evaluating the token-based system to find out whether it meets the specified requirements.** The system could be a simple tool or a full tokenized ecosystem, instantiated as one or more smart contracts or even L1 blockchain networks.

As a starting point, let's recognize that we are talking about *dynamical systems*. A *failure* is when the dynamical system gets stuck in a region of state space that does not reflect design intent: it's a *dynamical system fault*. It's like getting caught in the wrong loop.

There could be failures on the *logic* side; this is a *digital* problem. There could be failures on the *incentives* side; this is an *analog* problem. There could be failures on the combination; this is a *mixed-signal* problem. Fortunately, other engineering disciplines have similar challenges, and experience solving them. Here's how I think about it, with comparison to Electrical Engineering (EE).

Type of TE Verification	Type of TE Design	Verify What?	Where?	Parallel in Circuit Examples
Digital Verification	Digital: Discrete time (clocked), discrete-valued signals (typically binary).	Digital behavior, instantiated in **smart contract logic.**	Single smart contract, or a set of smart contracts.	Multiplexers, floating-point units, ARM cores
Analog Verification	Analog: Continuous time, or continuous valued-signals	Analog behavior, instantiated in smart contracts **incentive design** / economic model and other analog signals.	Single smart contract, set of smart contracts, or system level (may be across >1 chains).	Amplifiers, filters, memory bitcells.
Mixed-Signal Verification	Mixed-signal: Some digital and some analog blocks. Overall system is analog.	Mixture of analog and digital behavior.	Set of smart contracts, or system level.	Analog-to-digital converters, RF transceivers, memory columns

Given that the blockchain world has a decent initial handle on digital verification of smart contract logic (via formal verification), in this article I discuss:

- How to verify incentives? More generally, how to approach **analog verification** of token-based systems?
- How to verify both smart contract logic and incentives at once? More generally, how to approach **mixed-signal verification** of token-based systems?

Summary of Tools

I see three groups of tools emerging:

1. ***Human-based verification.*** Example: Le Grand Jeu board games.
2. ***Software-based verification.*** Example: cadCAD and TokenSPICE simulators.

3. *Economics-based verification / risk management.* Example: Kusama "canary" network for Polkadot.

Human-based verification is lowest effort and lowest fidelity. This is a good place to start. Software-based is higher effort but higher fidelity. And economics-based takes the most effort but has the most accuracy; it's a good place to end up. Let's review each in more detail.

Tool 1: Human-Based Verification

The main idea here is to leverage humans to vet the design. There are more obvious and less-obvious ways to do this. Let's review a few.

TE optimization-like methodology. This TE article section 4.3 suggested:

- (i) formulate the problem in terms of objectives & constraints
- (ii) try to fit an existing TE pattern to it, and
- (iii) design something new, if needed.

We can write (i) as a checklist. Then, "human" vetting is asking yourself how well you do against the checklist. One can use a TE Canvas (like this) to formalize the checklisting.

Feedback on whitepapers. In the 2017 heyday of ICOs, whitepapers were *the* key tool to communicate one's token design. But they can be used for verification too! First, the act of writing forces you to clarify your thinking. You can brainstorm and document possible failure modes, then try to address them (or explicitly ignore them). Second, you can send early drafts

to a handful of friendly experts for feedback. As those experts become more satisfied, you can share with an ever-increasing circle of people.

TE meetups. This takes whitepaper-style feedback and compresses it into a <20% of the time, with 80% of the benefit. At a TE meetup, someone presents their token design to a friendly audience in 10–20 minutes. The audience then tries to find flaws, and offer ways to improve the design. The presenter and audience collaborate to improve the design. Sometimes a whole new design emerges! Here's an example for a decentralized identity system.

TE Community Review. The TE community now offers a review process, where people can submit their prospective designs. See tokenengineering.org for more information.

Role-playing. When Joe Costello was running Cadence (a $B software company) in the early 1990s, he would regularly gather executives into a room for a special game. Each person would role-play a company in the industry. Surprising dynamics could emerge. Cadence management came to understand their competitors and collaborators better, because each player steel-manned the company they represented. It also unlocked creative new ideas for Cadence. Joe's leadership was so respected that Apple considered him to replace Steve Jobs.

Board games. Here, Token Engineering meets Dungeons & Dragons. Le Grand Jeu is "a hands-on tool to co-design sustainable micro-economies empowered by crypto-currency" [ref]. "When Le Grand Jeu met Token Engineering, it was love at first sight" [ref]. Below an example from a game played by the TE community in November 2019. Interestingly, it took courage to break free of "spreadsheet thinking" in order to use such human-

centric games for TE verification. Besides Le Grand Jeu, the TE community has riffed on Monopoly to explore cooperation schemes and more.

Tool 2: Software Verification

How does one design and verify a chip containing 10B transistors in under 3 months? *Software tools* are critical. It's so important that there's a whole industry — a $10B one — for software tools to design, simulate, and verify chips. Section 6 of this article illustrates tools for Electrical Engineering. Token Engineering (TE) needs similar tools.

Simulation tools, aka simulators, measure performance metrics of a given design (*controllable* variables) in a given environmental context

(*uncontrollable* variables). An uncontrollable variable may be (a) a *range* where the design must perform well at any of the variable's values, or (b) a probability distribution.

Verification tools verify that a design can work according to its performance metrics, despite uncontrollable variables. Verification tools can use simulators in-the-loop, as in the example image below. There are other in-the-loop possibilities too.

Results of a verification tool, to verify that a memory circuit hits its target yield of > 4 sigma. The uncontrollable variables follow a random probability distribution. The tool uses simulator-in-the-loop. The image is from this book. The technology is used in Solido Design Automation Inc (now a division of Siemens AG).

Given that we're still in the early days of TE, this article will focus on simulation tools. (Dedicated verification tools will come; all in due time.)

For TE, agent-based simulation (aka agent-based modeling, or ABM) is a good starting point. In agent-based simulation, there's a bunch of entities — agents — running around. Each agent has its own sensors, model of the world, decision-making framework, resources, and actions based on decisions. These agents can be super-stupid, like just doing stuff randomly. Or less stupid, such as using their model of the world to choose actions maximize their expected resources.

How do we actually *do* agent-based simulation? Here are three variants, with increasing fidelity:

1. **Spreadsheet-based agent-based system.** Each row is a different time step. Some columns are state variables; some are input variables; some are output variables. The next row's state variable values are a function of that row's input variables and the previous row's state variables. Output variables are a function of the current row's state variables and input variables.
2. **Custom software for agent-based modeling, with rough-grained models.** The rough grained models may be subroutines, differential equations or other "behavioral models". This approach takes more up-front effort than (1), but offers more flexibility. Once that up-front effort is invested, it's also easier to maintain and test, towards building more complex models.
3. **Custom software for agent-based modeling, with fine-grained "smart contracts in the loop".** This is even higher resolution than (2). Simulation time is longer but it starts to go with shades of gray into real-world behavior. It's akin to hardware-in-the-loop simulation.

(1) can be done by anyone who knows spreadsheets. This is great news, as it means that low-fidelity agent-based simulation can be done by many people.

(2) and (3) takes dedicated software. Netlogo is popular for (2); there are many others. For TE specifically, **cadCAD** is currently the most popular software. It's "an open-source Python package that assists in the processes of designing, testing and validating complex systems through simulation" [ref]. It allows one to go from lower fidelity (2) to higher-fidelity (3), and is tuned for TE problems.

Screenshot from cadCAD [source].

Tool 3: Economics-based Verification / Risk Management

So far we've discussed human-based and software-based verification, which are tools used in many engineering fields. The *economic* aspect is a unique property of token-based systems, and we can take advantage of it.

Here are two general approaches:

- Tool 3a. Ratchet up value-at-risk over time.
- Tool 3b. Give each person optionality in risk-vs-reward.

The next subsections explore each approach.

Tool 3a: Ratchet Up Value-At-Risk Over Time

Here's the recipe:

1. Launch the system live, with a small amount of value at risk (skin-in-the-game). Potentially allow centralized intervention.
2. As time passes or the network matures, increase the amount of value at risk. This can be done automatically or manually.
3. Eventually, there will be "full" value-at-risk. There can't be opportunities for centralized intervention.

In short:

> *"Bake slowly."*
>
> — *David Holtzman*

This pattern is already widely used in token-based systems. Here are some examples:

- **Built-in bug bounties.** When Bitcoin launched, it had security flaws that allowed double-spends and more. BTC had negligible value. But as more people discovered Bitcoin, BTC gained in value, and more energy was spent to fix Bitcoin's security vulnerabilities. To this day, if anyone finds the right flaw, they could make millions. Bitcoin has built-in bug

bounties. From a verification standpoint: Bitcoin ratcheted up value-at-risk over time. We can also call this "security bootstrapping" or "grow security" culture.

- **Explicit bug bounties.** Gnosis DutchX started with a testnet, giving a bounty of $150K for finding major security flaws. A flaw was found. Gnosis fixed it, then launched a second testnet. No flaws were found this time, then the mainnet was launched.
- **Emergency fixes.** Shortly after deployment, Spankchain was hacked; however the team quickly did an emergency fix and the project was saved. Some people gamed Balancer's liquidity mining program in its early days, so a quick fix was introduced. When MakerDAO lost its peg to the USD, people proposed emergency fixes.
- **Canary Networks.** Web3 Foundation launched the Kusama "canary" network, with 1% of the value of the eventual Polkadot mainnet. Cosmos' Game of Zones and Game of Stakes series each had skin-in-the-game testing out incentives.

Canaries were brought into coal mines to detect carbon monoxide sooner than miners. [Image CC0]. Sentinel species generalize upon canaries for detecting other dangers.

Here are more tactics. At first, they appear to be simply about being ready for emergencies. But ultimately, they more about scaling trust to a broader community ("progressive decentralization") as the project matures.

- **Emergency switches; upgradeable contracts.** Initially, Ethereum had a kill switch that a handful of people could trigger; with time this was removed. Many ERC20 tokens have "pausable" functionality, controlled by the founding team, which halts trading. More generally, many smart contracts are configured to be upgraded using e.g. OpenZeppelin. Call this "smart contract governance". While this gives short-term adaptability and fast response to hacks, it is vulnerable to the handful of people that control the contract or chain.
- **Explicitly *no* emergency switches; minimize governance.** The idea here is to *not* have any control once the contract or chain is deployed. It's a simpler system and avoids the centralized-control attack vector, but has less adaptability in the event of an emergency.

The previous two points appear like an either-or dilemma. However, there are a few ways to resolve it: (i) if your system is simple enough, just deploy it without governance, like Uniswap; (ii) have the emergency switch in the early days then remove it, like Ethereum; or (iii) spread control to more people over time, like MakerDAO or Compound.

The idea of "decentralize over time" generalizes beyond emergency switches. Here are related tools:

- **Web of Trust.** Here, the idea is to start with a small group of people that trust each other, then add more over time from existing members' networks. Social norms enforce subjective guidelines. MetaCartel Ventures is doing this in the investment space. Commons Stack is doing

this for open-source TE code with its "trusted seed". Finally, the "trust is risk" idea combines web of trust with staking: you're putting skin-in-the-game on people that you bring into the community.

- **Permissioned to Permissionless.** xDAI, Lukso, and Ocean Protocol all started as permissioned Proof-of-Authority (POA) and are moving to permissionless.
- **Centralized to Permissionless.** Compound started with control by a centralized team, and with release of its COMP governance tokens has delegated control to the community. Balancer is on a similar path.

Tool 3b: Give Each Person Optionality in Risk-vs-Reward

This is the classic risk-reward tradeoff. If a person takes more risk, they get potentially more reward. We can engineer this into token-based systems. Here are examples:

- **Every investment, ever.** Investing in an early-stage project brings more risk than, say, BTC or ETH.
- **Layer 2 General.** Layer-2 technologies like Lightning, Raiden, Plasma, ZK Rollups, and more provide benefits of speed, capacity, cost, or privacy but with potential tradeoffs to security. Over time, these technologies continue to soften the harshness of the tradeoff.

Economics-based verification tools have strong overlap with "risk management", hence the alternate subtitle for the section.

"Verification" versus "Validation"

Just because you've verified a TE design doesn't mean that it will meet the customer's needs. That's a question of *validation*. Validation is about knowing

whether you've hit product-market fit (PMF). Here are a couple comparisons:

- Validation asks "am I building the right product" and verification asks "am I building the product right?" [Ref].
- "Validation is the process of checking whether the specification captures the customer's needs, while verification is the process of checking that the software meets the specification" [Ref]. There are countless other comparisons.

This article focused on verification. However, some tactics it described can also be considered tactics for validation, like the TE Canvas. This article recommends achieving PMF before full decentralization.

Overall, aim to build the right product *and* to build the product right.

Conclusion

In this article, I explored tools for Token Engineering verification. I presented them in increasing order of fidelity. They can be used roughly in that order as well. That is:

- **Start with human-based verification.** Write down what you're aiming for, and write down your design. Do a TE workshop or play a game. Vet the design with friendly experts.
- **Then do software-based verification.** Spreadsheet out your design, and give cadCAD a whirl.
- **Do economics based verification / risk management.** Do one or more heuristics to *ratchet up the value-at-risk over time*. ***Bake slowly.***

And, don't forget to build something people want. Good luck and happy Token Engineering!

Acknowledgements

Thanks very much to these people for feedback: Julien Thevenard, Anish Mohammed, Cem Dagdelen, Angela Kreitenweis, Sebnem Rusitschka, Michael Zargham, Cyprien Grau, Griff Green, Sarah Vallon, and Monica Botez.

Video Version

The following video is from my Jul 19, 2023 EthCC talk "Token Engineering Verification: from TokenSPICE EVM Simulation to AI-Powered CAD".

- Related links: [**tweetstorm**] [**Video**] [**Slides — Gslides**] [**Slides — PDF**]

Related Reading

- Token Engineering introduction articles I, II, III.
- Specific TE articles: TE governance, and novel TE building blocks like Layered TCRs, and Curated Proofs Markets.
- TE website. It links to many more excellent articles by the burgeoning TE community.
- This post is a precursor to Ocean V3 work.

Updates:

- Sep 2020: added link to Ocean Protocol V3 posts
- Nov 2021: added link to TokenSPICE simulator.
- Jul / Aug 2023: gave an EthCC talk on this content (and more); and added "Video Version" section here

Follow **Ocean Protocol** via Twitter; chat with us on Telegram or Discord; and build on Ocean starting at our docs.

Blockchain | Defi | Ethereum | Homepage | Token Engineering

Written by Trent McConaghy

Following

7.1K Followers · Editor for Ocean Protocol

Trent McConaghy. @OceanProtocol , AI, data, Web3. www.trent.st

More from Trent McConaghy and Ocean Protocol

Trent McConaghy in Ocean Protocol

Mission & Values for Ocean Protocol

Democratizing data while retaining privacy rights

6 min read · Jul 23, 2018

1.3K 3

Ocean Protocol Team in Ocean Protocol

Ocean Protocol Update || 2024

Accelerate Predictoor, C2D Springboard, Ocean Enterprise

10 min read · 4 days ago

329

Manan Patel in Ocean Protocol

Where to Stake your OCEAN

How to make the most of your OCEAN? Read our guide to staking, liquidity provision,...

4 min read · Jun 2, 2021

192

Trent McConaghy in Ocean Protocol

Token Engineering Case Studies

Analysis of Bitcoin, Design of Ocean Protocol. TE Series Part III.

9 min read · Mar 2, 2018

872 3

See all from Trent McConaghy

See all from Ocean Protocol

Recommended from Medium

Nillion

Next on Nillion: The Nucleus Builders Program

Nillion is on a mission to decentralize high value data across a wave of new web3 use...

3 min read · Jan 19, 2024

70 1

dreamsofdefi in across.to

The intents bridge: Cross-chain value transfer and the future of...

Tldr; As the cross-chain ecosystem has evolved, many bridges have emerged to hel...

7 min read · Nov 2, 2023

224

Lists

data science and AI
40 stories · 92 saves

My Kind Of Medium (All-Time Faves)
67 stories · 235 saves

Modern Marketing
84 stories · 463 saves

Ramsès Fernàndez-València in Innovation Stories

Homomorphic Signatures

Approaching the topic and the main proposals

11 min read · Sep 21, 2022

Pontem Network in Pontem Network

What is Friend.tech—the SocialFi sensation on Base?

All about the viral SocialFi app on Base | friend.tech airdrop | Share pricing & fees | To...

14 min read · Oct 28, 2023

213

57

Crypto Fairy in Coinmonks

Under the hood of zkSNARKs — PLONK protocol: Part 1

PLONK is one of the zero-knowledge proving systems that belong to the SNARK group....

8 min read · Nov 3, 2023

26 1

Deborah Simpier in Althea

Unlocking Telecom Assets

Trillions of dollars of telecom assets, left underutilized as a result of legacy systems,...

5 min read · Oct 24, 2023

15