



First Audit Platform for Smart Contracts

Whitepaper by Eduard Kotysh, Rob Hitchens and Alex Rysenko.

© 2017 Solidified Technologies, Inc.

Contents

Contents	2
Executive Summary	3
Why Ethereum Smart Contracts?	4
The Need for Review	4
The Need for a Crowd	6
The Need for Transparency	6
Use case 1 - HKG Token Reissue	6
Use case 2 - ICO Bounty Programs	6
Use case 3 - TheDAO Hack	7
Contracts, Developers and Reviewers	8
Spec of Intended Behaviour	8
Workflow	10
Bounty Payment	11
Finalization	11
Reputation	11
Review Process	12
System Architecture and Development	12
Security Considerations	13
Gas	13
Integration with Github	13
Scalability	14
Testing and Rollout	14
Product Roadmap	15
Future Outlook	16
Appendix	18
Dispute Resolution	18

Executive Summary

Solidified is a crowd sourced platform for screening smart contracts on the Ethereum Network, with a goal of improving the quality of the Ethereum ecosystem in a transparent manner.

This document describes the purpose of this community and how it operates, as well as the underlying systems that support it.

With the increasing popularity of blockchain technology across various industries, the stakes of developing bug-free smart contracts are high. The existing option of performing a professional audit is expensive, the results are not transparent to the public, and even multiple-firm audits don't always yield results.¹

There is no affordable path to a dependable code audit with multi-party peer review. Solidified aims to provide a collaborative web platform where developers can recruit qualified reviewers to assist with meticulous code review. Solidified is built on the familiar principles of transparency, incentivization and the wisdom of crowds.


Solidified incentivizes code review by offering rewards to reviewers who discover and report defects in code submitted for review. Solidified provides a strong signal of confidence (or skepticism) to those who, themselves, lack the time or skills to form independent opinions about smart contract quality and suitability for purpose.

For code authors, Solidified provides:

- Access to pool of ready, willing and qualified code reviewers.
- Valuable feedback and the opportunity to mitigate issues before they emerge as serious problems in post-deployment scenarios.
- Independent assessment of the quality of their work.
- An alternative to expensive third party code audit services that may not catch every important defect.
- A wider range of review perspectives as compared to the private audit process.

For code reviewers, this provides:

¹ Lessons from the DAO incident,
<http://www.rsk.co/blog/lessons-from-the-dao-incident>

- 
- A steady supply of (potentially) paid work opportunities. Authors offer bug bounties that are payable upon discovery of defects.
 - An opportunity to gather reputation in a forum that focuses on non-trivial issues.

For investors, executives and non-technical stakeholders:

- Strong signaling from specialists who understand the nuances of smart contract design and implementation.
- Confidence in the quality of contracts upon which the success or failure of enterprises ultimately rests.
- Standardized metrics.

For the Ethereum community as a whole:

- Significantly raises the quality-assurance bar which should lead to increased confidence in adoption of the Ethereum ecosystem as a whole.

Why Ethereum Smart Contracts?

Ethereum is the most popular smart contract platform in the rapidly expanding field of blockchain technology².


Ethereum's core value proposition is driven by smart contracts, programs which execute agreements between previously untrusted parties. They aim to automate and improve efficiency in a wide range of industries, all without a need for central authority, but with guaranteed execution.

The Need for Review

The excitement around smart contracts arises in no small part from their unique ability to bind actual assets with unstoppable and incorruptible logic. This novel and powerful form of software is attracting great attention from all levels of society, significant investment and experimentation, and developers are keen to wade in and learn how they work.

Smart contracts govern assets and are often entrusted with significant value. Some contracts govern large assets, while others protect the functionality of highly sensitive processes, such as

² Cryptocurrency Market Capitalization, <https://coinmarketcap.com/#USD>



Etherplan, which manages people's investments ³, or Bitland, which manages real estate title transfer in Ghana. ⁴

Needless to say, this puts a lot of responsibility on the developer to write bug-free code that works as intended and, importantly, repels malicious users who want to make it work in unintended ways. Given the immutable nature of smart contracts, non-trivial defects and security holes may be difficult or impossible to correct. Developers are therefore tasked with adapting to unfamiliar paradigms, learning new languages and patterns and producing defect-free applications.

Given that even minor defects can result in catastrophic and irreversible consequences, smart contracts call for an abundance of caution.

Additionally, consider the following:

- Decentralized code is difficult to write and test for, even for seasoned developers
- The Ethereum community generally recognizes that errant smart contracts (and resulting losses and controversy) damage the technology's reputation. There is widespread awareness of the need to mitigate this risk.
- The authors do not believe that the technology is too dangerous to use safely. Rather, there is an urgent need to develop safety procedures.
- Thus, a reputable trust source should emerge.
- This platform should be available to all, with a low barrier to entry.

Solidified aims to fill the role of such reputable review platform.

Our review process incorporates:

- A large variety of reviewers of different levels
- Cheap cost of entry (bug bounty tiers)
- Visibility of results, i.e. what recommendations were actually implemented

Solidified will provide an important quality signal to non-technical stakeholders such as shareholders in enterprises that may succeed or fail based on correct smart contract execution. Solidified will do so by attracting the attention of expert specialists and *incentivizing* their careful inspection and reporting.

³ Etherplan, <https://etherplan.com/>

⁴ Bitland, <http://www.bitland.world/about/>



The Need for a Crowd

Professional audit firms exist for smart contracts ([Zeppelin Solutions](#), [New Alchemy](#), [Bok Consulting Pty](#)) but the cost is high and unattainable to an average developer. Some even resort to raising VC funds just to pay for the audit in successfully jump-starting an ICO.

An alternative program has emerged as a de facto standard practice. Bug bounties offer lucrative rewards for major issues found. Case in point Gnosis, a prediction service smart contract, which launched two rounds of bug bounty, with rewards reaching \$50,000 for major issues found. ⁵ There is still no single go-to place where Reviewers can learn about bounties on offer and there is no normalized set of terms, i.e. the rules of the contest. This places a very high barrier to entry for developers who wish to commence a bug bounty program.

Solidified aims to be the first platform to provide a lower barrier for entry by connecting the blockchain developer community in a peer review format.

The Need for Transparency

Solidified wants the results of the review process to be visible to the community. Without this, the community cannot trust the contract. To highlight this, we present several use cases:

Use case 1 - HKG Token Reissue

Even after a public code audit by Zeppelin, ether.camp's HackerGold Token (HKG) was found to have a bug serious enough to warrant a reissue of the token. ⁶


The code audit recommended changes that would have prevented the bug in the HKG token contract. Clearly there is a difference between the recommendation and actually correcting flaws in the code. To know whether or not the the results of the review have been implemented, the patches need to be visible.

Use case 2 - ICO Bounty Programs

It is now standard practice for companies to offer a percentage of their token supply for bug bounty during the ICO, showing the need for crowd-based review.

⁵ Gnosis Bug Bounty Round II \$50k bounty for severe bugs,
<https://blog.gnosis.pm/gnosis-bug-bounty-round-ii-50k-bounty-for-severe-bugs-fcc855b55cc>

⁶ Ether.Camp's HKG Token Has A Bug And Needs To Be Reissued,
<https://www.ethnews.com/ethercamps-hkg-token-has-a-bug-and-needs-to-be-reissued>



CryptoPing ran an ICO from May 25th to June 24th, allocating 8,000 bounty tokens for identifying bugs and issues.⁷

Crypviser offered 3% of tokens for all bounty campaigns collected during the ICO.⁸
InvestFeed ran an ICO July, 23, 2017 to August 7, 2017, allocating 1.5% of tokens for the rewards and the bounty system.⁹

Use case 3 - TheDAO Hack

On June 17, 2016 TheDAO was subjected to an attack that exploited a combination of vulnerabilities, including a known bug related to recursive calls, and the user gained control of 3.6 million Ether, valued at about \$50M at the time of the attack. TheDAO had been independently audited and placed into production before an Ethereum developer on GitHub pointed out a flaw relating to "recursive calls". This flaw was later exploited in the attack.

It's worth noting that even after the defect was well-understood *it was not possible to amend to the contract* to resolve the issue. Instead, the protocol and chain history was amended in a controversial public debate. This radical solution will be inaccessible to the majority of smart contract developers.

This is noteworthy because it underscores the importance of getting contracts right before they are released, the scale of losses that are possible in case of error or oversight and the limitations of relying on a single source of quality assurance.

⁷ CryptoPing Concludes Its ICO, <https://themerkle.com/cryptoping-concludes-its-ico/>

⁸ Crypviser Bounty campaigns, <https://medium.com/@crypviser/crypviser-bounty-campaigns-d4f6bd456b13>

⁹ InvestFeed launches ICO, <https://www.coinspeaker.com/2017/07/21/1investment-network-investfeed-to-launch-an-ico-on-july-23/>



Contracts, Developers and Reviewers

There are 4 main entities on the Solidified platform:

Contracts - the solidity code that's being reviewed. Contracts are posted on Solidified by developers and screened for bugs by reviewers. Contracts have a reputation of trust associated with it based on ratings received, and a public discussion board for the registered developer community to comment on.

Issues - the bugs discovered in the contract by Reviewers. We categorize them into major and minor bugs found, where major bugs represent severe issues that must be fixed before the contract can be marked Solidified, and typically carry a higher bounty.

Developers - the blockchain programmers who want their contract reviewed. You become a developer by registering on the site and validating your email. The barrier for entry is intentionally set low, to encourage all levels to be part of the community.

Reviewers - the trusted experts of the community who have been verified by Solidified to hold an adequate level of expertise in Ethereum and Solidity. This includes blockchain programmers who've been working in the industry for at least 2 years, as well as Certified Ethereum Developers (CEDs) by B9Lab or C4.

While in our MVP, we take a centralized approach to screening the pool of Reviewers and assuring high quality, in subsequent releases we plan on leveraging reputation as means of earning the Reviewer rank.


Spec of Intended Behaviour

To help relay the intended behaviour of the contract, we introduce the Spec of Intended Behaviour (SIB) as a new standard for relaying the intended operation of smart contracts between the author and a reviewer. SIB adds a form of explicit intent to the otherwise purely implicit intent of the code.

In a nutshell, SIB is a structured document that Developer fills out, spelling out the conditions that should hold during contract execution, identifying which industry best practices¹⁰ and security considerations¹¹ have been covered (and not covered), and what testing has been

¹⁰ ConsenSys: Smart Contract Best Practices <https://github.com/ConsenSys/smart-contract-best-practices>

¹¹ Ethereum Security Considerations <http://solidity.readthedocs.io/en/develop/security-considerations.html>



performed on the contract thus far (i.e. what percentage of code coverage was achieved with unit tests, what testing was performed on the TestNet, etc).

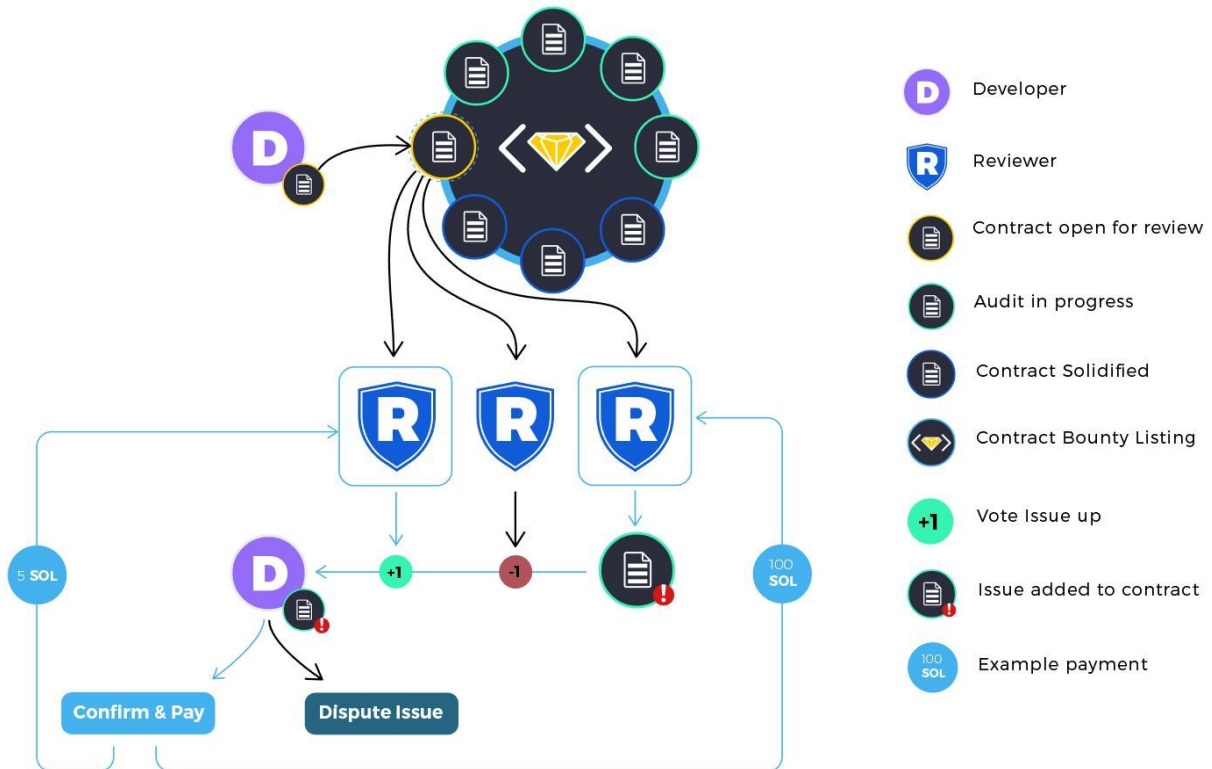
In order to easily communicate the intended behaviour SIB relates several important pieces of information about the Smart Contract:

- Conditions and Assumptions that should hold during contract execution
- Circuit breaker condition
- Gas estimation
- Upgrade Path
- Testing History
- Re-use of existing audited components (i.e. OpenZeppelin Framework)
- Rollout Plan
- All known issues and limitations
- Further details as we progress through our detailed design process.

Workflow

1. Developer posts contract(s) and Spec of Intended Behavior with Bounty. Bounty funds are held in escrow. Solidified has no privileged access to these funds as they are held in an Ethereum smart contract.
2. Solidified provides a familiar off-chain discussion board organized around the contract under review, where people can suggest improvements, answer each other's questions, point out common oversights, and get a sense for how the contract is viewed in the public eye.
3. Reviewers may claim a bounty and upload supporting documentation to illustrate why their claim is valid. Such evidence may include written explanation, unit test exposure, screen captures/imagery or even video demonstration of the fault.
4. Additionally, reviewers are invited to vote in support or against the claim. That is, they lend credibility to the claim, or vote to dismiss it as invalid/inconsequential.
5. If the Developer acknowledges the claim and does not file a dispute within a preset grace period (current thinking is this should be approximately 7 days), then the bounty is automatically awarded.

Astute readers will note the presence of community consensus about the importance of reported bugs, and the presence of unresolved disputes in step 5.





Bounty Payment

Any issues filed against a contract are subject to bounty payout set by the developer.

In order for the Reviewer to get the bounty, one of two things must happen:

- 1) Developer must acknowledge that the issue is valid, or
- 2) The dispute resolution process must find in the Reviewer's favor.

If either of those conditions holds and the total bounty available has not been exhausted, the Reviewer gets paid. Importantly, Reviewers will have a line-of-sight into Bounty funds remaining before they report a bug.

Finalization

In case of disagreement (step 5, above) we turn to the wisdom of the crowd to settle the matter. In summary, we attract more attention to the issue and incentivize qualified Reviewers to help determine if the bug report is valid. For more information, see "Dispute Resolution" in the Appendix.


Reputation

Solidified aims to be a reputable source of smart contract quality signals and expert ethereum reviewers.

To accomplish this, we've added two rating systems which measure 1) the level of trust people have in the contracts and 2) the reputation of reviewers.

Contracts establish reputation largely through the existence of unclaimed bug bounties and weighted upvote/downvote surveys.

Anyone can quickly locate popular contracts, see the level of expert confidence, peruse the reported issues and investigate the Developer response with links to patches that have been made. Voting is weighted by Reviewer reputation that is earned by merit.



Reviewers establish reputation by being reliable. Successfully claiming a bounty adds to reputation (largest prize), as does correctly assessing the validity of a bug claim. Reputation is bound to upvote/downvote actions solving for “nothing at stake”.

Review Process

The process starts with the developer uploading or linking a contract to Solidified for review and filling out a Spec of Intended Behaviour. This spec helps communicate what the contract is intended to do and acts as ground zero of truth for our review process. See “Spec of Intended Behavior” for more details.

When Reviewer initiates the audit, he/she first goes through the following steps:

- 1) Review the Spec of Intended Behaviour to understand the conditions under which the contract should operate, existing limitations and the testing history.
- 2) Review the Code of the contract to assess whether the intended behaviour is observed, guided by the checklist of common smart contract vulnerabilities.
- 3) If an issue is found, lay a claim to the bounty by creating an Issue ticket, describing the details of the concern and providing the necessary supporting deliverables (unit test exposing the issue, demonstration screencast of Remix browser, etc.)

Once an issue is filed, a 7-day grace period begins, during which the Developer can contest the validity of the issue filed (see Dispute Resolution). If no contest is made within the grace period, the Reviewer gets the bounty.

System Architecture and Development

The Solidified web platform is being developed using a traditional MEAN stack (MongoDB, Express.js, AngularJS, and Node.js) on a highly secured AWS microservices architecture.

We chose it to take advantage of the full-stack JavaScript development (team’s expertise), flexible NoSQL database schema and powerful front-end UX tools. It is a proven set of technologies adopted by PayPal, Uber, DowJones and many other big players in the production environment that have shown to scale successfully to the masses and deliver a great user experience.

We chose AWS to be our cloud host provider due to team’s deep experience with launching multiple startups on Amazon Web Services stack and its reputation of being the market leader in public cloud computing.



Stages of Development include:

1. Prototype (completed)
2. Escrow contract implementation (completed)
3. Dispute resolution contract (in-progress)
4. Setup of a secure and scalable AWS Cloud environment for Development, Stage, Production (in progress)
5. Implementation of dockerized API microservices (in progress)
6. Front-end implementation (in-progress, mid-Oct '17)
7. Beta test with a subset of signed up users (early Nov '17)
8. Public Launch (late Nov '17).

Security Considerations

Solidified is aimed to provide a painless platform for smart contract review prior to publishing to the Ethereum network. To accomplish this, we are building a combination of service-based stack, web-based user interface and a set of audited smart contracts backing escrow process and dispute resolution.

Our backend is enclosed in AWS VPC with only API endpoint exposed to outer world through AWS ALB. Internal services use specific AWS IAM accounts with limited permissions. API endpoint uses common API Gateway pattern and additionally implements rate-limiting on the balancer level to protect against basic service overuse. To protect against more complex attacks like DDoS we're relying on CloudFlare, one of the largest DDoS protection networks in the world.

Gas

One of our technical concerns is reducing the gas cost associated with the smart contracts used by the Solidified platform. By minimizing the cost overhead, we maximize the accessibility of Solidified platform. We will be evaluating every aspect of system functionality and will be reasoning boundary between code that's executed in smart contract and code that's executed on our backend or frontend thus achieving best security without sacrificing accessibility (that could be lowered by high gas costs).

Integration with Github

We plan to tightly integrate with Github to utilize their API and make use of existing issues, pull requests and commenting functionality on Github.



Scalability

Scalability challenge is different for Solidified platform backend and on-chain part of the system. For backend we will be using classic horizontal and vertical scaling approaches, dynamically introducing more server nodes when needed by leveraging AWS auto-scaling policies and Application Load Balancers (ALB) for routing the traffic efficiently. For on-chain part, scalability is largely dependant on the underlying Ethereum protocol, however we will do our due diligence to assure efficient gas cost throughput on the Ethereum network. Most optimizations here will lie in simplifying our contracts without sacrificing security of them.

Testing and Rollout

Testing is critical for Solidified platform and will be done on multiple levels. Both, individual micro-services and smart contracts will be covered with unit tests. Interaction between components will be covered by unit and integration tests. Once the platform is deployed to development and staging stacks, automatic continuous integration and functional testing will be executed.

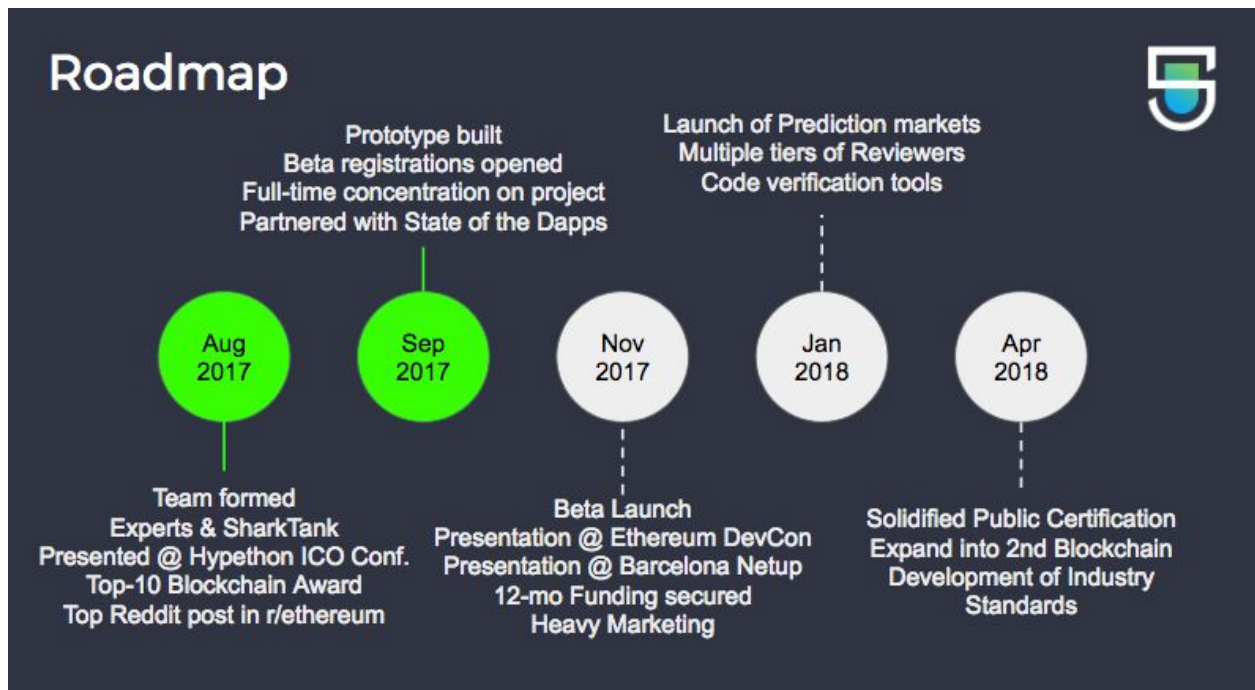
Smart contracts will be first deployed to the test Ethereum network and tested by us until we verify the correctness of intent. After our preliminary testing is complete, we will publish our contracts for public review on our site. Besides posting bug bounties, we plan on adding a simple voting system on the issues discovered to get a better signal from the developer community as to which issues require higher priority attention.

In this spirit of transparency, backed by the power of the crowd, we want to give everyone a chance to take a look at our code before we deploy it to production stack and the live Ethereum network.

Our select group of blockchain experts from our network has agreed to test-drive the platform on the test net, and will be giving us valuable feedback prior to public release.

Product Roadmap

We divided our Product into: MVP, V2 and the Final Vision, in order to gather the community around a working platform early, while keeping an eye on the grand vision we aim to achieve.




Beta Launch - **Nov, 2017**

The goals for the first version of our Product are:

- To create a Bug bounty platform for Smart Contracts with a user-friendly UX.
- To create a community of Ethereum Developers and Startups on our platform
- To expand our vetted network of Top Blockchain Experts

V2 - **Jan, 2017 (+90 days)**

The next release adds a formal audit process targeted at startups that have a lot of money at stake. This is a premium service available to token holders only, that provides a formal verification process (not a bounty hunt) of smart contracts signed off by top experts. Other features include adding a subjective signal to our Platform. We leverage predictive markets to vote on whether the community believes this contract is worth investing in, or likely to fail.



Vision - April, 2018 (+180 days)

Our Vision milestone is about creation of a Certification process, a Universal Standard of Quality for Smart Contracts, determined by the crowd, that can be used to gauge the quality of any contract on the blockchain.

- To establish a public certification process (Solidified Stamp) based on technical review of the contract and the subjective prediction signals.
- To promote and adopt this certification stamp in the industry
- To expand into other non-Ethereum based Smart Contract blockchains
- To expand the services category by not only offering a marketplace for bug bounties, but also for fixing the issues that were found.

Future Outlook

After the value of crowdsourcing contract audits has become ubiquitous in the developer community, our intention is to continue developing new products that can create more advanced interactions between users on the Solidified Platform and its growing community, such as providing automated verification tools, a service for fixing bugs, and Premier Access to highly reputable reviewers.

Automated Verification Tools

Incorporate code verification tools into our platform, and make them available for developers and reviewers to find issues quicker.

Contract Development as a Service

A key feature of the growing community will be to allow reviewers that identify bugs to be paid to fix the issues and become contributors to the project being audited. This workflow will enable Solidified users to create project groups, learn new skills and offer their developer services for a contract fee or negotiated sum.

Reviewer Tiers

In order to attract experts of various level, and to provide a better selection for developers, we plan on introducing tiers of reviewers, where level of expertise carries a corresponding bounty range. From the low hanging issues with entry-level tier, to an in-depth analysis by the popular personas in the blockchain community who act as our VIP reviewers.



Template Marketplace

As the blockchain technology grows and common themes of contracts start to emerge, we would like to provide a database of customizable templates that have been audited and accepted by the community as trusted base to build upon. This will help new developers adhere to a set standard of best practices and not re-invent the wheel.

We feel these future developments will help us become a one stop shop for smart contract services and help the community raise the quality bar of the Ethereum ecosystem.



Appendix

Dispute Resolution

When a developer takes issue with a claim and refuses to authorize release of the bounty, we offer the Reviewer an opportunity to accept this outcome or object to it. The Reviewer is required to place a non-trivial amount of funds and reputation at stake. The issue is then handed to an arbitration process that turns greater community attention to the issue and the evidence.

This process draws ideas from opinion markets. In summary, the community are invited to place wagers (something at stake) on the ultimate outcome of the community review that will find in favor of the Developer (wants to dismiss the bug report), or the Reviewer (wants to be paid for the report). Importantly, the Developer will also have more at risk than the original Bounty as a disincentive to disingenuous initial actions leading to this sort of dispute.

This pattern of grace periods and the option of accepting the decision or escalating the dispute continues through two additional tiers with increasing risk at each level. The rising costs are analogous to the procession of traditional courts, starting with small claims court and leading ultimately to the “Supreme Court” *if the parties involved cannot accept outcomes and wish to place more at stake and push the matter to a higher level.*


Arbitration mechanisms such as this have been implemented on the blockchain, and we may leverage one of these trusted platforms in order to not to reinvent the wheel and save time.

Opinion-based Metrics by Vote

The design team is formulating incentive-based voting for metrics that do not lend themselves to a final true or false (all or nothing) result. For example, questions like “Is the code readable?” and “Is the code well simplified for the task?” don’t lead to a single true answer. Even so, many useful signals will fall into this category. Consider, 90% a case of qualified developers reporting the code is unreadable and more complex than it needs to be. That would be a non-confidence signal.

The design aims to include the following broad requirements in the final version:

1. Solve for “nothing at stake”. Voting is to be consequential action with rewards and penalties bound to correctly assessing the situation. Something at stake will help mitigate voter apathy and encourage participation.

- 
2. Possibility of “winning” by taking a contrarian viewpoint. That is, Solidified will incentivize voting against the evident majority. This will mitigate trend-following reflexive voting with the crowd behaviors that would otherwise distort the signal.
 3. Ultimate resolution. Issues must be ultimately settled in order to determine the awarding of prizes. The finding will likely be determined by a panel of qualified experts. The precise panel formation and decision-making process is under design at this time. Candidate inputs include user reputation, ETH holdings and re-use of the bug bounty voting module.