A Trading Market for Prices in Peer Production

MALVIKA RAO, Incentives Research
GEORG J.P. LINK, University of Nebraska at Omaha
DON MARTI, Mozilla
ANDY LEAK, Mountain View Smart Contracts
RICH BODO, Mountain View Smart Contracts

1. INTRODUCTION

Open source software forms much of our digital infrastructure [Eghbal 2016]. Peer production, the mechanism behind open source software development is an organizational innovation where individuals, in a diverse and distributed community, self-match to the tasks best suited to them [Benkler 2002]. Peer production has successfully tackled complex, uncertain projects, underlying billions of dollars in software production [Open Source Press Release 2015]. However, this digital infrastructure is increasingly under strain, facing escalating demand for services and a lack of adequate resources to address them [Eghbal 2016]. Vulnerabilities in software have been exploited, attracted public attention, and caused large financial damages. Figures on software errors and cyber crime are estimated to lie in the billions [NIST 2002; Dreyer et al. 2018]. Earlier research foresaw these types of problems. Kooths *et al.* [Kooths et al. 2003] assert that the absence of price signals in open source development means that users' valuations remain unknown. Hence, supply and demand do not fully align. Anderson argues that information insecurity is partly due to a failure in the design of incentives [Anderson 2001].

In fact, software quality seems to be below the level preferred by users and developers alike. That is, users are willing to pay to avoid the risks of using broken software, and developers are willing to fix software if compensated. Thus, there appears to be a failure of market design, and an opportunity to create better market mechanisms that incentivize a higher quality equilibrium in software production. This leads to the research question that motivates our work:

> *How can a market design incorporate price signals into peer production, facilitate information sharing, and promote quality?*

In this paper we make progress towards answering this question. We present preliminary work on a *futures trading market* for finding and fixing software issues. Participants in our market can create futures contracts to predict whether these issues will be addressed, hedge the risks to which they are exposed by using defective software, and incentivize work. They can trade on questions such as: will a bug be fixed? will a vulnerability be found? Our platform is not restricted to coding tasks – rather it can be used for various purposes (e.g., information discovery, code reviews, testing, documentation).

Bug bounty programs are the state of the art for reporting vulnerabilities. Open source bounty systems offer rewards for reporting and fixing bugs. Marketplaces for software tasks include crowdsourcing platforms for bug bounties (e.g., Bugcrowd [BugCrowd Inc. 2012], Hackerone [Hackerone 2012]) and open source bounties (e.g., Bountysource [Bountysource Inc. 2013]), crowdsourcing contests (e.g., Topcoder [TopCoder Inc. 2001]), and crowdfunding sites (e.g., Kickstarter [Kickstarter 2009]). However, these market mechanisms have limitations that we address with a novel market design based on trading futures contracts. Research on software economies and vulnerability reporting has advocated a market-based approach but with a different focus than the present paper [Bacon et al. 2009; Bacon et al. 2010; Rao et al. 2014; Schechter 2002; Ozment 2004; Hosseini et al. 2012].

2. OUR APPROACH

We propose a futures trading market for eliciting information and incentivizing tasks.

Example 2.1. User Adam documents a bug in an issue tracker (now identified as bug $\#1337$) and goes to the futures trading market. Adam creates a *fixed* offer with a maturation date in two weeks for a payout of $\$200$. Adam buys 200 units—$1 potential payout each—at a unit price of $\$0.8$, paying $\$160$, by depositing the money into escrow. Developer Beth sees the *fixed* offer, has time to fix bug $\#1337$ within two weeks, and decides to accept the offer. Beth buys the 200 units at a unit price of $\$0.2$, paying $\$40$, by depositing the money into escrow. The contract is now formed: Adam owns the *unfixed* position and Beth owns the *fixed* position. Two weeks later, on maturation, if bug $\#1337$ is fixed then the issue is closed. Beth earns the reward of $\$160$ and gets her $\$40$ deposit back. If bug $\#1337$ is unfixed then the issue remains open. Beth loses her deposit while Adam receives his and Beth's deposits, earning $\$40$.

The above example illustrates a simple case of how such a market might work. A contract is associated with outcomes which can be verified in an issue tracker. On maturation, the contract pays out to the owner of the position corresponding to the issue's status. Our design departs from existing market mechanisms in several ways. Current approaches to software work typically reward the reporter of a vulnerability or a fix. However, software is often developed in a collaborative fashion, where a final solution builds upon the input of others [Howison and Crowston 2014]. Further, a task may require different areas of expertise from different individuals. Existing mechanisms do not seem to have a way to assign credit to all contributors. This limitation is addressed by a trading market because a participant may do partial work on a contract and resell his position (see Example 2.2).

Open source bounty systems have extra transaction costs of claiming bounties, and fail to incentivize meta work since rewards must be explicitly divided among multiple testers and developers instead of letting the system handle it. Bug bounty programs that reward the disclosure of vulnerabilities do not capture users' valuations for fixes. Unlike prediction markets, we have a large number of contracts but a small number of participants per future contract ("wisdom of individuals" revealed to crowds of users). Participants in a prediction market typically cannot influence the outcome whereas our futures draw participants who have information about that issue and thus may affect the outcome.

Why do we call it a futures trading market? Similar to futures contracts, there is a quoted price in the market place for the expectation that an issue will (not) be closed at a specified future (maturation) date. The price of entering a contract is equal to zero but a deposit into escrow is required. At any time before maturation, the owner of a contract can sell his position and receive the difference in price since he entered the contract and the deposit back. On maturation, the owner of a contract pays with his deposit for the outcome or receives his deposit and that of the counter-party.

2.1 Design features

Example 2.2. As before (Example 2.1), Adam ($\$160$ for the *unfixed* position) and Beth ($\$40$ for the *fixed* position) enter into a futures contract worth a payout of $\$200$ in two weeks depending on the status of bug #1337. One week later, Beth realizes that she does not have the expertise to fully fix bug $\#1337$. Beth decides to submit a partial fix and sells her *fixed* position. Charles buys the 200 units of the *fixed* position from Beth at a unit price of $\$0.4$, paying $\$80$ to Beth. Beth had paid $\$40$ and receives $\$80$, earning $\$40$ for her partial work. On maturation, if bug $\#1337$ is fixed then the issue is closed. Charles earns the reward of $\$200$ for a net gain of $\$120$. If bug $\#1337$ is unfixed then the issue remains open and Charles receives nothing but loses the $\$80$ paid to Beth, while Adam receives his and Beth's deposits from escrow, earning $\$40$.

Partial work. Because the market allows developers to earn credit for partial work, it incentivizes information sharing and facilitates collaboration (see Example 2.2).

Dependencies. The completion of task A may depend on task B being completed first. A developer who has bought the *fixed* position of a contract on task A (with the intention to do the work himself) might then create a new offer to pay for the completion of task B, with an earlier maturation date.

Multiple contracts. It is possible to create multiple contracts for the same issue (e.g., multiple contracts that all offer to pay for a fix for bug $\#1337$), where the maturation dates and payment amounts may (not) be the same. The *fixed* positions on these contracts may (not) be owned by the same person in order to accumulate rewards for doing the work.

Indemnity. Inspired by dominant assurance contracts [Tabarrok 1998], our design allows for the possibility of an indemnity being paid if a request is not fulfilled. With this market design, a belief that something cannot happen is handled the same way as an incentive to make it happen. Aggregating all such contracts, the market potentially creates a pool of wealth that can be captured by innovators.

Decoupling funding from work. The payout from a futures contract is solely dependent on the status of the issue in the issue tracker. At maturation the owner of the position corresponding to the issue's status (e.g., fixed or unfixed) is paid, regardless of who may have done the work to resolve the issue. This enables scenarios where investors may put up the capital and hire a team to do the work.

2.2 Proof-of-concept implementation of the futures trading platform

We validate our design through a proof-of-concept implementation [Hudson and Mankoff 2014], called *Bugmark*. The benefits of using the Ethereum blockchain are anonymity of market participants, an immutable, decentralized, and audit-able record of transactions for trust, and automation via smart contracts. The code is written in HTML, CSS, JavaScript, and Ruby, and uses a PostgreSQL database.

Futures Contract Design. We describe the basic elements of a Bugmark futures contract (underlined) by walking through the life of such a contract from creation to maturation, reflecting the process in example 2.1. A new contract gets formed when two buy offers match. To form a contract, two buy offers—one for a *fixed* position, the other for an *unfixed* position—have to match in price, volume, statement, and maturation date. The unit price of the buy offers must together equal to $\$1.00$ per unit. For example, respective unit prices of $\$0.20$ for a *fixed* position and $\$0.80$ for an *unfixed* position. The total price for each user is equal to the unit price multiplied by the volume. Thereby, the two buy offers together pay the money into escrow that will be paid out on maturation. On the maturation date, the statement (e.g. 'bug $\#1337$ is fixed') is evaluated to determine who gets the payout from escrow. When the statement is true, the owner of the *fixed* (otherwise, *unfixed*) position gets the payout.

Validation through simulation. Large multi-user systems, such as Bugmark, are challenging to design. The system is non-deterministic and evolutionary because the user experience depends on the decisions of other users. Using agent-based-modeling [Ren and Kraut 2014], we carry out a simulation[1] to demonstrate that Bugmark works and enables data collection to evaluate design changes. In this simulation, agents have no knowledge of the environment and randomly choose to open buy offers for fixed and unfixed positions. The agents use the full price range from 0.00 to 1.00 and choose volumes between 30 and 50 units. With these parameters, there is good likelihood that offers match and enter into contracts. As positions are formed, the number of open contracts decreases. Over time, the number of contracts reaches five because agents only trade on five issues with the same maturation date. As more offers match, however, the volume of positions on each contract increases. With this infrastructure built, we can simulate various agent behaviors and test hypotheses about the characteristics of the system. In ongoing work, we are designing experiments with real software teams.

[1]The simulation code is available under an open source license online: https://github.com/bugmark/bmx_bots

3. CONCLUSION AND FUTURE WORK

This paper proposes a novel market design for software development in peer production communities, based on futures trading markets. The core of our innovation lies in introducing price signals in such a way as to leverage and strengthen the successful qualities of peer production. By enabling developers to earn credit for partial work, our market incentivizes information sharing and collaboration. The market treats a prediction that something cannot happen in the same way as an incentive to make it happen. Thus, the market creates a pool of wealth that can be captured by innovators. Several promising lines of research stem from this work. How does our model compare to other models? Does a trading market enable new types of tasks and what is a measure of *task complexity*? What is the impact of introducing price signals? What is the relationship between markets and peer production?

REFERENCES

Ross Anderson. 2001. Why Information Security is Hard – An Economic Perspective. In *Proceedings of the 17th Annual Computer Security Applications Conference (ACSAC)*.

David F. Bacon, Eric Bokelberg, Yiling Chen, Ian A. Kash, David C. Parkes, Malvika Rao, and Manu Sridharan. 2010. Software Economies. In *Proc. FSE/SDP Workshop on Future of Software Engineering Research*.

David F. Bacon, Yiling Chen, David C. Parkes, and Malvika Rao. 2009. A Market-Based Approach to Software Evolution. In *Proc. 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications (OOPSLA '09)*.

Yochai Benkler. 2002. Coases penguin, or, linux and the nature of the firm. *Yale Law Journal* 112, 3 (2002), 369–446.

Bountysource Inc. 2013. Bountysource Inc. website. *https://www.bountysource.com*. (2013).

BugCrowd Inc. 2012. BugCrowd Inc. website. *https://bugcrowd.com*. (2012).

Paul Dreyer, Therese Jones, Kelly Klima, Jenny Oberholtzer, Aaron Strong, Jonathan William Welburn, and Zev Winkelman. 2018. *Estimating the Global Cost of Cyber Risk: Methodology and Examples*. RAND Corporation, Santa Monica, CA.

Nadia Eghbal. 2016. *Roads and Bridges: The Unseen Labor Behind Our Digital Infrastructure*. Ford Foundation.

Hackerone 2012. Hackerone website. *https://www.hackerone.com*. (2012).

Hadi Hosseini, Raymond Nguyen, and Michael W. Godfrey. 2012. A Market-Based Bug Allocation Mechanism Using Predictive Bug LifeTime. In *Proc. 16th European Conference on Software Maintenance and Re-engineering (CSMR)*.

James Howison and Kevin Crowston. 2014. Collaboration Through Open Superposition: A Theory of the Open Source Way. *MIS Quarterly* 38, 1 (2014), 29–50.

Scott E. Hudson and Jennifer Mankoff. 2014. Concepts, Values, and Methods for Technical HumanComputer Interaction Research. In *Ways of Knowing in HCI*, Judith S. Olson and Wendy A. Kellogg (Eds.). Springer, New York, NY, 69–93. DOI: 10.1007/978-1-4939-0378-8_4.

Kickstarter 2009. Kickstarter website. *https://www.kickstarter.com*. (2009).

Stefan Kooths, Markus Langenfurth, and Nadine Kalwey. 2003. Open-Source Software - An Economic Assessment. *MICE Economic Research Studies* 4 (2003).

NIST. 2002. *The economic impacts of inadequate infrastructure for software testing. Planning report 02-3*. http://www.nist.gov/director/planning/upload/report02-3.pdf.

Open Source Press Release 2015. The Linux Foundation Releases First-Ever Value of Collaborative Development Report. https://www.linuxfoundation.org/press-release/the-linux-foundation-releases-first-ever-value-of-collaborative-development-report/. (2015).

Andy Ozment. 2004. Bug Auctions: Vulnerability Markets Reconsidered. In *Third Workshop on the Economics of Information Security*.

Malvika Rao, David C. Parkes, Margo Seltzer, and David F. Bacon. 2014. A Framework for Incentivizing Deep Fixes. In *Proc. AAAI Workshop on Incentives and Trust in E-Communities*.

Yuqing Ren and Robert E. Kraut. 2014. Agent Based Modeling to Inform the Design of Multiuser Systems. In *Ways of Knowing in HCI*, Judith S. Olson and Wendy A. Kellogg (Eds.). Springer, New York, NY, 395–419. DOI: 10.1007/978-1-4939-0378-8_16.

Stuart E. Schechter. 2002. How to Buy Better Testing: using competition to get the most security and robustness for your dollar. In *In Infrastructure Security Conference*.

Alexander Tabarrok. 1998. The private provision of public goods via dominant assurance contracts. *Public Choice* 96 (1998), 345–362.

TopCoder Inc. 2001. TopCoder Inc. website. *https://www.topcoder.com*. (2001).