Compliance Adherence in Distributed Software Delivery: a Blockchain Approach.

Kapil Singi, Pradeepkumar D S, Vikrant Kaulgud, Sanjay Podder
Accenture Labs, Bangalore, India
{kapil.singi,p.duraisamy,vikrant.kaulgud,sanjay.podder}@accenture.com

ABSTRACT

In this extended abstract, we propose a conceptual framework that leverages distributed ledger technology and smart contracts to create a decentralized system to capture the occurrence of interesting development activities (e.g., a development build) and associated contextual data, and automatically audit and evaluate compliance to governance policies. Our hypothesis is that such a framework will facilitate easier sharing of information across all participants of a distributed development team, compliance evaluation and early mitigation actions, leading to greater visibility and compliance. Currently, the proof of concept we are working on is focused on sharing and compliance evaluation of the open-source components used in software development.

CCS CONCEPTS

• **Software and its engineering** → *Software creation and management*;

KEYWORDS

Distributed Teams, Compliance evaluation, Regulations, Open-source, Distributed Ledger Technology, Blockchain, Smart contracts

ACM Reference Format:
Kapil Singi, Pradeepkumar D S, Vikrant Kaulgud, Sanjay Podder. 2018. Compliance Adherence in Distributed Software Delivery: a Blockchain Approach.. In *ICGSE '18: ICGSE '18: 13th IEEE/ACM International Conference on Global Software Engineering , May 27–29, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3196369.3196383

1 INTRODUCTION

Software reuse [1]; for example, the reuse of open-source components [2],[3], enables faster software development with reduced time, effort and costs [6],[7]. The prevalence of open source usage is highlighted in a report published by analyst firm Gartner where it mentions that the usage of open-source components forms 50%-80% of software code [1]. However, uncontrolled or unmonitored acquisition and usage of open-source components poses risks pertaining to non-compliant licenses and security vulnerabilities [4],[5]. For example, a crowd worker may acquire (i.e. download) a Math library from an open-source component repository like Maven, not realizing that the library's GPL license violates the open-source policy of the enterprise. Industry-specific regulations like the Sarbanes-Oxley regulation also impose stringent requirements on the internal controls for information security. Use of non-compliant or vulnerable open-source components not only cause software quality issues, but more importantly could cause regulatory exposure as well.

The other factor that aggravates this concern is that software development is becoming increasingly distributed, with small teams and crowd workers forming distributed software development teams. These teams work in their own, siloed development environments where they may use different methods and tools for constructing software. They may reuse third party components during development, and may also share their work with other teams as part of a larger collaborative development environment. This environment resembles a software supply chain, where there are many participants including development teams and component repositories collaborating and adding value to create the final software asset. IT services companies or enterprises for which such teams work have reduced control over the tools and methods used by them. This loss of control and visibility raises critical questions related to quality of compliance to governance policies and regulations, tracking of work progress etc. For example, the use of non-compliant open-source Math library but may not be reported by the crowd worker to the enterprise. If such non-compliance is detected late in the development life cycle, it may lead to unplanned defect fix effort and time. The worst scenario is that if remains undetected cause poor software product and hefty fines due to regulatory exposure. Further, because the distributed teams and crowd workers may never have worked together, there is a lack of trust between these participants, hampering information sharing and willingness to participate in audits etc.

The Compliance Platform for Distributed Software Development attempts to address the above issues. We use a distributed ledger technology / blockchain and smart contracts as the infrastructure for creating a decentralized environment that will be used to:

- Store software development events of interest (for example, a development build), and contextual data (for example, when was the build triggered, which crowd worker triggered the build, which open-source components are part of the build etc.)
- Evaluate compliance of these events and data to governance policies and regulations.
- Notify the distributed development team participants of non-compliances, and advise them of best practices and mitigating actions.

[1] https://www.gartner.com/doc/3730417/managing-digital-trust-software-development

ICGSE '18, May 27–29, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5717-3/18/05…$15.00
https://doi.org/10.1145/3196369.3196383

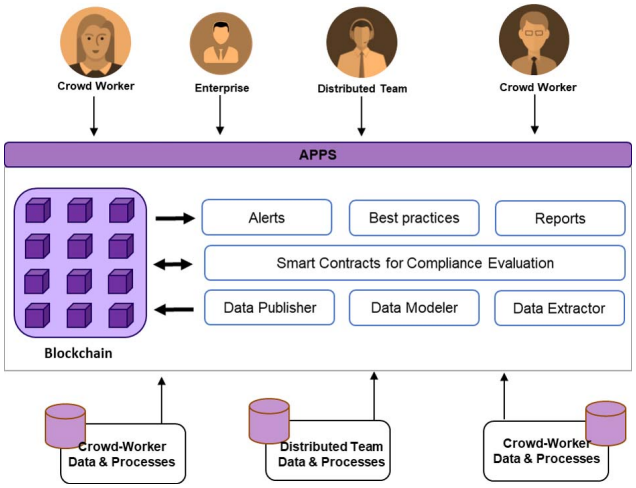

Figure 1: Compliance Platform Architecture

2 COMPLIANCE PLATFORM FOR DISTRIBUTED SOFTWARE DEVELOPMENT

Blockchain or a distributed ledger technology is a decentralized data store that stores cryptographically secured and linked records of transactions or activities between multiple parties (or blocks of information). The technology provides immutable blocks; i.e., once a block is created, the information in it cannot be modified or deleted, nor can the block be deleted. Further, it uses various consensus algorithms to ensure that all blockchain participants validate and agree to the injection of a new block in the blockchain. This makes the blockchain virtually tamper-proof. Finally, the blockchain technology provides for decentralized computation or code execution through 'smart contracts'. These smart contracts can process the data in existing blocks as well as external (off chain) data and execute various rules and transactions.

Given the nature of issues related to visibility, compliance and trust in a distributed software development environment, we hypothesize that blockchain and smart contract technology would provide a single unalterable instance of development data from across the distributed team and the necessary compliance evaluation techniques for a trusted and compliant development environment. We also take inspiration from software engineering and non-software engineering use-cases such as naming and storage system [8], health records sharing system [9] and crowdsourcing platform for posting and receiving tasks [10].

Figure 1. illustrates the architecture of the Compliance Platform. The platform uses event monitors to listen for events in the development environments used by distributed teams and crowd workers. The event occurrence and related data is first converted to a common format so that participants can share data from disparate environments like Maven, Gradle etc. This data is augmented with additional information required for compliance evaluation. For example, a build event data is augmented with license (GPL, MIT, BSD etc.) and security vulnerability information from CVE or NVD for the libraries used in the build. This augmented data is injected in to the blockchain for creating a new, immutable, secured block representing the build event.

In the second step, as soon as a new event block is created, Compliance evaluation smart contracts start executing. The Compliance platform comes with a library of smart contracts, one for each governance policy and regulatory compliance. For example, the license compliance smart contract analyzes the license information in the build event block and triggers the creation of a new compliance block in case the build used a non-compliant open-source license. This way, the blockchain starts the immutable and secure recording all build events in the distributed development environment and any non-compliances. All participants of the blockchain can query these blocks any time to ascertain the state of work and quality of compliance leading to better visibility, auditability and trust.

The platform also provides for advisory and reporting features. The advisory features provide alerts to non-compliant participants so that they can take mitigation steps. Reporting allows an enterprise to quickly understand state and quality of work.

3 FUTURE WORK

We have implemented a proof of concept to show compliance to open-source license and security vulnerabilities policies. As a future work, we intend to pilot this in real-life distributed projects to ascertain the technology challenges and measure the benefits. We will also determine the optimal operational characteristics of the solution âĂŞ who owns the platform, how we manage the permissions etc. Finally, we will also introduce a reputation score for incentivize governance policy and regulation compliant software development.

REFERENCES

[1] Krueger, C.W., 1992. Software reuse. ACM Computing Surveys (CSUR), 24(2), pp.131-183.
[2] Kogut, B. and Metiu, A., 2001. OpenâĂŘsource software development and distributed innovation. Oxford review of economic policy, 17(2), pp.248-264.
[3] Mockus, A., Fielding, R.T. and Herbsleb, J.D., 2002. Two case studies of open source software development: Apache and Mozilla. ACM Transactions on Software Engineering and Methodology (TOSEM), 11(3), pp.309-346.
[4] Bhoraskar, R., Han, S., Jeon, J., Azim, T., Chen, S., Jung, J., Nath, S., Wang, R. and Wetherall, D., 2014, August. Brahmastra: Driving Apps to Test the Security of Third-Party Components. In USENIX Security Symposium (pp. 1021-1036).
[5] Haddox, J.M. and Kapfhammer, G.M., 2002. An approach for understanding and testing third party software components. In Reliability and Maintainability Symposium, 2002. Proceedings. Annual (pp. 293-299). IEEE.
[6] Tu, Q., 2000. Evolution in open source software: A case study. In Software Maintenance, 2000. Proceedings. International Conference on (pp. 131-142). IEEE.
[7] Androutsellis-Theotokis, S., Spinellis, D., Kechagia, M. and Gousios, G., 2011. Open source software: A survey from 10,000 feet. Foundations and TrendsÂő in Technology, Information and Operations Management, 4(3âĂŞ4), pp.187-347.
[8] Ali, M., Nelson, J.C., Shea, R. and Freedman, M.J., 2016, June. Blockstack: A Global Naming and Storage System Secured by Blockchains. In USENIX Annual Technical Conference (pp. 181-194).
[9] Azaria, A., Ekblaw, A., Vieira, T. and Lippman, A., 2016, August. Medrec: Using blockchain for medical data access and permission management. In Open and Big Data (OBD), International Conference on (pp. 25-30). IEEE.
[10] Li, M., Weng, J., Yang, A., Lu, W., Zhang, Y., Hou, L. and Liu, J., 2017. CrowdBC: A Blockchain-based Decentralized Framework for Crowdsourcing. IACR Cryptology ePrint Archive 2017: 444.