A Data Extraction Methodology for Ethereum Smart Contracts

(Oral Communication)

Flavio Corradini[1], Alessandro Marcelletti[1], Andrea Morichetta[1], and Barbara Re[1]

University of Camerino, Camerino, Italy
{flavio.corradini, alessand.marcelletti, andrea.morichetta, barbara.re}@unicam.it

The Ethereum blockchain is gathering interest thanks to its key features, including security, transparency, and decentralisation. In such a context, the extraction of data generated by the execution of blockchain-based applications is useful to support their continuous improvement. Indeed, thanks to smart contracts, decentralised applications are executed directly in the blockchain, generating data that can be used for certified auditing and monitoring activities [1, 2, 3]. Among the different analysis techniques, process mining is an emerging solution for analysing blockchain applications exploiting data (i.e., logs) resulting from smart contract executions [5]. Process mining is a set of techniques that can be used to identify bottlenecks and deviations from the expected behaviour of the monitored processes [7]. In particular, the process mining community is moving toward object-centric event data representing the backbone of managing and analysing complex process data [8]. The blockchain context can benefit from such an analysis but still lacks a suitable extraction methodology to process the complexity of data generated by blockchain-based applications. Considering blockchain applications, the execution of a smart contract generates data that is stored in blocks (e.g., timestamp), transactions (e.g., sender, inputs, gas, and more), events and storage (i.e., the memory containing the smart contract state). Additional effort is also required to decode information that cannot be easily interpreted in its original form in the blockchain. Thus, the extraction activity has to deal with the heterogeneity of storage and decoding factors. Moreover, catching the state changes of a contract permits a comprehensive understanding of the application and enables detailed analysis of the contract evolution over time. Differently from transaction and block, a state change does not generate a clear and accessible track, requiring a deep investigation of the low-level data structure [4, 9]. In Ethereum, each variable influencing the state of a smart contract is permanently stored and encoded in the storage memory based on a specific slot. In the case of simple variables, this slot is statically assigned, while for complex types (e.g., mappings and structs), this is dynamically combined with a key generated during the execution. In the last few years, some approaches were proposed to extract data stored in different blockchain sources [6]. However, these approaches mainly extract information related to the execution of smart contract functions (e.g., events, inputs, senders) without considering the evolution of its state.

For these reasons, we propose **a data extraction methodology to extract data from Ethereum smart contracts including execution-related data and state changes**. The proposed methodology is divided into some steps described in the following. The first step retrieves the **contract code** starting from the *contract address* and the *contract name* from which to extract data. Then, all *contract transactions* from which data is extracted are collected. Once the smart contract source code is obtained, we **compile the contract** to obtain three particular outputs: (i) Application Binary Interface (ABI), (ii) Abstract Syntax Tree (AST), and (iii) storage layout. This information is used to decode extracted data. The **extraction of contract state** data is the focal point of the proposed methodology, and it extracts the state

variables updated after each contract execution. For this purpose, each transaction is replicated on a local environment with the state of the blockchain at the moment in which the transaction was originally executed. This returns the transaction trace containing the list of executed operations (i.e., opcodes) and the state of the EVM (i.e., memory locations). By analysing such operations and their inputs/outputs, it is possible to reconstruct the history of state variable changes and retrieve their location in the case of dynamic ones. As a result, the step produces the *storage state* that is later included in the final log. Once the contract state changes are collected and decoded, the methodology **extracts transactions and blocks data**. For each of them, the methodology takes the name of the executed function from the corresponding log and its inputs, decoded thanks to ABI. Similarly, the methodology **extracts events** emitted by smart contract functions and contained in the transaction log. Using the ABI, events of past transactions are captured together with the name and the value of the attributes. The last step of the methodology **creates the JSON log** containing all the extracted data. This log is stored and provided to the user, who can use it for different analysis scopes.

To demonstrate the feasibility of the proposed solution, the methodology was implemented as a web application that makes the data extraction of a smart contract accessible by taking the contract details from the user as input. Also, we tested the benefits of our methodology using the PancakeSwap Ethereum smart contract, but it can be generally applied to any Ethereum smart contract.

References

[1] Claudio Di Ciccio, Giovanni Meroni, and Pierluigi Plebani. Business process monitoring on blockchains: Potentials and challenges. In *Enterprise, Business-Process and Information Systems Modeling*, volume 387 of *LNBIP*, pages 36–51. Springer, 2020.

[2] Claudio Di Ciccio, Giovanni Meroni, and Pierluigi Plebani. On the adoption of blockchain for business process monitoring. *Softw. Syst. Model.*, 21(3):915–937, 2022.

[3] Flavio Corradini, Alessandro Marcelletti, Andrea Morichetta, Andrea Polini, Barbara Re, and Francesco Tiezzi. Engineering trustable and auditable choreography-based systems using blockchain. *ACM Trans. Manag. Inf. Syst.*, 13(3):31:1–31:53, 2022.

[4] Kiarash Diba, Kimon Batoulis, Matthias Weidlich, and Mathias Weske. Extraction, correlation, and abstraction of event data for process mining. *WIREs Data Mining Knowl. Discov.*, 10(3), 2020.

[5] Leyla Moctar-M'Baba, Mohamed Sellami, Walid Gaaloul, and Mohamedade Farouk Nanne. Blockchain logging for process mining: a systematic review. In *HICSS*, pages 1–10. ScholarSpace, 2022.

[6] Leyla Moctar-M'Baba, Mohamed Sellami, Walid Gaaloul, and Mohamedade Farouk Nanne. Blockchain logging for process mining: a systematic review. In *HICSS*, pages 1–10. ScholarSpace, 2022.

[7] Wil M. P. van der Aalst. Process mining: A 360 degree overview. In Wil M. P. van der Aalst and Josep Carmona, editors, *Process Mining Handbook*, volume 448 of *LNBIP*, pages 3–34. Springer, 2022.

[8] Wil M. P. van der Aalst. Object-centric process mining: Unraveling the fabric of real processes. *Mathematics*, 11(12):2691, 2023.

[9] Jochen De Weerdt and Moe Thandar Wynn. Foundations of process event data. In Wil M. P. van der Aalst and Josep Carmona, editors, *Process Mining Handbook*, volume 448 of *LNBIP*, pages 193–211. Springer, 2022.