UNSW
AUSTRALIA

School of Computer Science and Engineering

Faculty of Engineering

The University of New South Wales

A Web-Based Process Discovery Tool for Ethereum Applications

by

Shenghan Gao

Thesis submitted as a requirement for the degree of

Bachelor of Engineering in Software Engineering (Honours)

Submitted: Aug 2021

Supervisor: Dr. Christopher Klinkmüller and Dr. Dilum Bandara

Student ID: z5211215

Abstract

A blockchain is a decentralized digital ledger of transactions that is secured by cryptography. A smart contract is a digital protocol running on a blockchain that ensures security and efficiency. With the rise of the smart contract blockchain platform, there's an increasing demand for understanding user behaviour, monitoring of business processes, and detection of fraudulent actions. Thus, the analysis of smart contract transaction data is critical. This project developed a web-based analytics dashboard for using process discovery to analyse smart contracts that are deployed on the blockchain, in particular on public Ethereum. This tool supports a more efficient Ethereum data analysis process to meet the demand. This thesis begins with a background and literature survey, analyse the underlying technologies and actual use cases, and evaluate the contributions and drawbacks of each related work. The thesis then focuses on the design and implementation of the tool, with a detailed assessment and discussion of the results achieved, as well as the final conclusions and possible future works.

Acknowledgements

First and foremost, sincere thanks to my supervisors, Dr. Christopher Klinkmüller and Dr. Dilum Bandara, who gave me lots of professional guidance and constructive suggestions throughout the one-year thesis courses. Their insightful advices, valuable feedbacks and constant encouragement, which have inspired me and enriched my knowledge, not only in the professional area such as blockchain and process mining, but also in how to become a better engineer, I couldn't have finished the work alone without their help. Moreover, I would like to extend my gratitude to my accessor, Dr. Helen Paik, for her valuable advice, feedback, and generous support.

Secondly, the work also inspired by the numerous literature authors, programmers and experts that I have referenced in the thesis, unfortunately, I cannot thank them one by one here in spite of their great contribution to this work.

Lastly, I would like to thank my parents and friends who have been giving me great encouragement, they are the motivations that support me to go on. Only with their selfless support, care and love, can I overcome the difficulties and pursue my study til now.

Abbreviations

BLF	Blockchain Logging Framework
BPMN	Business Process Model and Notation
DApps	Decentralized Applications
DAG	Directed Acyclic Graph
DFGs	Directly-Follows-Graphs
ELF	Ethereum Logging Framework
EPC	Event-driven Process Chain
PoW	Prove of Work
P2P	Peer to Peer
P2SH	Pay-To-Script-Hash
SC	Smart Contract
UI	User Interface

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Blockchain is a distributed ledger that enables cryptographically secure transactions and that is immutable, transparent, decentralized, and secure. Second-generation blockchain platforms like Ethereum also brought the capability to store arbitrary data and allow for executing smart contracts, which are the user-defined source code. However, with the increasing transaction volume and use cases of blockchain platform users requires further analysis and control over the platform to maintain and monitor business process, to detect fraudulent events and to prevent illicit activities.

In this situation, analysing blockchain data is critical for helping with the above objectives. A typical analysis step is to firstly extract the raw data and convert it to an appropriate format , then apply analysis techniques with the transformed-data. There is a lot of literature on the specific analysis approach, there are also a lot of tools and techniques out there for each step, but overall the tools support are fragmented. Thus users need to extract data with some tools, apply transformation with others, and then load it into analysis tools. Hence our goal is to design and develop a tool that provides a web-based analytics dashboard for using process discovery to analyse smart contracts that are deployed on the Ethereum, which integrates all tools, speed up and ease the analysis process. This thesis focuses on Ethereum due to its popularity and smart contract feature, but also focus on process discovery as it has been demonstrated to be a useful toolbox, for example in the case of using it to analyse CryptoKitties (Klinkmüller et al., 2020).

Chapter 2 explains the background on related technologies and project description, including the aim and the motivation of the project, as well as the usage scenarios and problem statement. Chapter 3 provides a summary of the relevant literature, along the proposed solution and the similar applications. Chapter 4 gives the tool design including the software design and rational of decisions of each iteration. Chapter 5 addresses the Implementation details such as architecture and techniques used. Chapter 6 addresses and evaluates the current outcomes. Finally, Chapter 7 draws up conclusions and the future work.

Chapter 2

Background

In this chapter, we address the background knowledges such as Blockchain (2.1), particularly, Ethereum (2.1.1). Moreover, Process Mining (2.2), and Usage Scenarios and Project Description (2.3) also addressed, including the problem statement, aim, motivation and significance of the project.

2.1 Blockchain

Blockchain is an append-only distributed ledger that can store data. It has the following features, which make it special compared to others. The first one that it is decentralised, meaning it does not rely on a central point of control. Instead, it relies on consensus protocols across a network of nodes to confirm any transaction performed on the network. In this scenario, participants on the network all must agree unanimously to add a new block and must do it while ensuring its integrity (Nakamoto, 2008). The second important property is immutability, it refers to how data remains unchangeable once it has been recorded and processed on the blockchain, meaning it is also protected from any modifications or attacks. Consensus mechanism is used to ensure the blockchain system under distributed architecture. One block can be proposed into the chain, depending on which node first obtains the right and is accepted by most other nodes. It is not solely decided by one node, but the final consensus of most nodes, which ensures data consistency. The process of altering data on a block requires control many nodes in the network, which is impossible to do so and leads to the immutability of the blockchain. Each block also stores a hash of the preceding block creating a chain going back all the way to the first block created. Therefore, records are permanent and impossible to modify. Thirdly, blockchain makes data transparent. Network participants can access holdings and transactions of public addresses using a block explorer, used to search the blocks of a blockchain,

including their contents and relevant details. Moreover, blockchains are secured through a variety of mechanisms that include advanced cryptographic techniques and game-theoretic models of behaviour.

The blockchain's structure is why it has these special properties, that consists of blocks containing transaction information linked sequentially from the back to the front, as illustrated in Figure 1. Each block in the blockchain has a hash pointer to that block, and stores a hash pointer to the previous block, except for the first one (Fleder, Kester and Pillai, 2015). Such a data structure ensures that the data cannot be tampered with because once tampered with any block of data, the corresponding hash pointer will be wrong. So any small manipulation of existing data would cause a huge difference in the encrypted hashes, therefore ensures immutability and security.

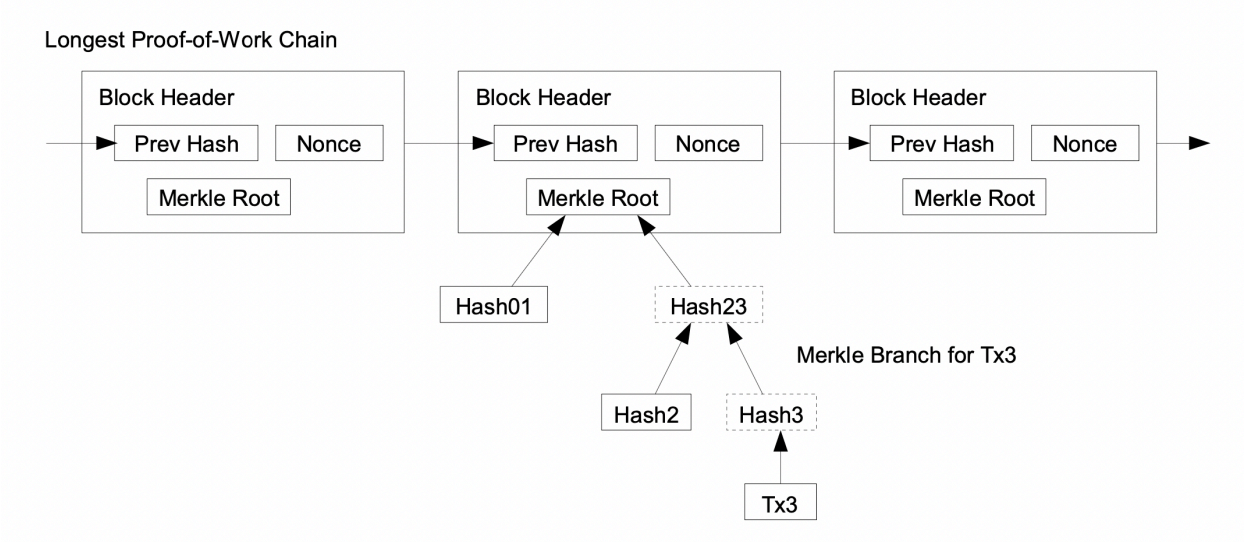

Figure 1. Blockchain Structure

The second feature of blockchain is that it's a peer-to-peer network, which is the communication mode between different nodes in the blockchain system and also a distributed network structure. With P2P networks, different nodes can interact directly with each other, and each node connected to each other is in a peer position. Each node acts as a server, providing services to other nodes as well as using services provided by other nodes, which ensures decentralization and anonymity of the platform.

Through the combination of direct P2P network, chain-like data structure and consensus mechanism to ensure that the data is reliable and immutable, thus realising a self-trust system that does not require an intermediary organisation, also simplifying the process and improving efficiency.

The first generation platforms typically only allow for executing pre-defined functionality, that is,

typically each transaction refers to a transfer of cryptocurrency from one account to another. Second generation platforms however allow to additionally host and execute arbitrary functionality, for example, Ethereum, can execute smart contracts, which is mainly focused on in the thesis. It has all the features that a blockchain platform has, also what distinguishes Ethereum from other mainstream decentralised platforms is developers can specify the conditions for their Dapps, and then the Ethereum network executes it, so-called smart contract. It can express triggers, conditions, and business logic to enable complex programmable transactions, deployed, invoked, and self-executed across the Ethereum network by all connected nodes (Chan and Olmsted, 2017). More details are reviewed in section 2.1.1.

2.1.1 Ethereum

Ethereum is generally considered the second most popular blockchain platform, But unlike most other blockchain platform, Ethereum aims to be more than just a means of exchanging and storing value. Besides storing the transactions data, Ethereum is also store the latest status of the smart contract and the code for each of it. As for the smart contract, which is simply a program that runs on the Ethereum platform, the most common language is Solidity. It can handle all the aspects of the contract, for example, enforcement, supervision, execution, and payment (Chan and Olmsted, 2017). To be noticed, smart contracts are also immutable, once it is deployed, it cannot be changed, therefore it would require that a programmer keeps a high level of accuracy and logical correctness when writing smart contracts.

Smart contracts can be used for many different scenarios, developers can build smart contracts that provide functionality for other smart contracts, much like how software libraries work. Alternatively, smart contracts could just be used for transparently automating businesses on the Ethereum blockchain. For example, if there's a smart contract that is used for fund-raising, it does not need a central organisation to actively collect the money from everyone who donates, the contract itself knows whether the money is sufficient or not and the amount that everyone sent. The fund would automatically be sent to the raiser if the requirement has met, otherwise no one can withdraw the money out of the pool.

Users have to pay for executing a smart contract, the cost is paid to nodes that consume memory, power, storage, and computing, and the unit of cost is called Gas. When users execute a smart contract, they must define the maximum amount of Gas to be consumed, otherwise if the execution process exceeds the amount of gas, it would be halting down. The expense of gas eliminates the

halting problems, makes the code for smart contracts more efficient and optimised. Also, gas usage reduction would also be an essential topic for people to analysing through transaction data.

There is one famous example of Ethereum Applications called CryptoKitties, which is a game allows players to purchase, collect, breed and sell virtual kitties, and all of these processes are based on Ethereum network and smart contract. As a representative decentralised application, the analysis of data in CryptoKitties would help us understand the specific user behaviour in the actual Ethereum application, also it contributes to inspire the study of blockchain data analytics. This thesis discusses the CryptoKitties analysing use case in the section 3.3.1.

2.2 Process Mining

Process mining is defined as a method of distilling a structured process description from a set of recorded events. Information systems, e.g., Oracle, capture the process data and recorded as events, process mining techniques take this data and convert it into an event log. This event log contains multiple pieces of key information that are critical. Process mining software is able to create a flowchart or model that lets users know the flow of the process, including the timing of each step and all the changes in the process by applying the event log. It is also a way to infer workflows and to effectively use the audit trails present in modern information systems (Van der Aalst, 2016). It has also become more important over the past decade, since it allows for a comparison with predefined desired models as well as monitoring changes from the standard process over time, and they can take actions to optimise performance by changing certain aspects of the used process activities.

In traditional business process management, process modelling or analysis is conducted through process workshops and interviews, and the result is an idealised description of the process. Process mining uses the existing data in the enterprise information system to automatically present the real process. The benefit of applying process mining algorithms is that the discovery and optimisation process is faster, more data-driven, and more complete, with less effort than a traditional process flowchart that can take a team few days to complete. Moreover, process mining also made the investigation processes on a large scale and at low labor cost across the company, at the same time analyse the process thoroughly and accurately based on facts, and it can be applied in variety of industries: financial services, telecommunications, manufacturing, health care or consumer goods and other industries.

Basically, the process mining has three types, first is process discovery, which creates models, discovers and visualises process models that reflect the behaviour. Second is conformance checking, compare a pre-defined or ideal process to the actual process, diagnose and quantify deviations by replaying the event data on the model. Then is a model enhancement, that is used to improve or extend a process model using event data, enriches it by additional information such as performance, cost, probabilities, or social structures (Müller and Ruppel, 2019).

In the thesis focuses on process discovery algorithm, to analyse data in Ethereum, which is one of the most challenging process mining tasks, which mainly focus on the discovery task and the control-flow perspective, constructed process model and reflecting the behaviour in the log. There are plenty of process discovery algorithms, such as InductiveMiner, AlphaMiner, DFG mining, etc.

There are examples that describe the steps of the process discovery technology which are commonly used today. Most of the process discovery tools take the essential data, and convert it into a sorted, formatted data-set for the actual analyse, the challenge is that the transforming of the actions of the users into a human-understandable events. Then, the process discovery tools create the process models. Based on the models, it would become obvious to show the detail information or to standardised the process.

Currently, there are tools and algorithms on the market that provide comprehensive functionality such as visualisation and modelling. Process discovery can be easily achieved by using some mature integrated software such as ProM, due to its convenience, most of the time one single event log file is needed as input to perform process discovery. Otherwise, some of the algorithms can also be implemented via other programming languages such as Javascript or Java, more technical detail is addressed in chapter 4.1 software design overview.

2.3 Usage Scenarios and Project Description

Adoption of blockchain technology is still in its infancy. Currently, a lot of financial companies, governments and businesses are trying to find the adaptability of blockchain technology in their respective areas. New type of business models or traditional business are expected to emerge, but so far there are few instances of significant applications in the production of blockchain systems in the

industry. However, in the foreseeable future, blockchain will definitely catalyse new business models in many important fields, such as production chain, financial services, logistics services, etc， which reduces the cost of building business relationships, and possibly reducing the cost of transactions.

Then, of course in this kind of situation, more data analysis is needed to make sure the process is safe and analyse user behaviour. There's an increasing demand for understanding user behaviour and for monitoring the process, in order to achieve this, analysing smart contract event data is critical. In order to analysis, for example security analysis, see if the code is working as expected from process perspective, or to improve the application based on how people are using it.

There are many fragmented tools with various functions, such as some tools can extract and transform data formats, some tools can apply process mining algorithms for analysis, and so on. However, in order to complete a whole analysis process, such as analysing a deployed smart contract, due to the limitations of the tool itself, the user may need to perform one sub-function through one tool, and then another sub-function through another tool, which is time-consuming and inefficient.

Therefore, our tool integrates all the tools, and it has these features:

- A user-defined way of extraction of data. In the tool the user can define how event data is extracted from the Ethereum blockchain. User should use the query language from Blockchain Logging Framework to perform extract function, in which user could compose their own query to specify the extracting criteria and check the validity of the query they've composed, more detail is addressed in section 5.2.1
- The extracted data is then applied by a process discovery algorithm and the result process model would be generated and visualised in the frontend user interface, more detail is addressed in section 5.2.2

Chapter 3

Literature Survey

In this chapter, we address the literature survey in which solutions and techniques with different approaches that contribute to blockchain data analytics are examined. Particularly, the survey deals with related technologies (3.1) for Extraction of Process Mining Data from Ethereum (3.1.1), Graph Analysis of Blockchain Transactions (3.1.2). Moreover, Related Application of Blockchain Analysis Platform (3.2) also addressed, including BlockSci (3.2.1) and Analysis of Business Process on Blockchain Infrastructure (3.2.2).

3.1 Technologies for Extracting Event Data and Visualising the Process

In general, the expected outcome is to have a tool that support the analysis process, and there's two steps, the first is the data extraction function, the raw data of blockchain is not suitable for process mining, there is a mismatch between the logged data and the event data required for analysis, and Blockchain Logging Framework (BLF) is the tool that relief the pain, allows users to query Ethereum in more flexible ways, retrieves data from Ethereum block by block in historical order, and extraction of data into formats suitable for analysis, and we expected to integrate the functionality into the tools to extracting process event data from Ethereum.

Secondly, the tool are expected to have basic analytical functions, to have a structured process model from a set of events data, users should be able to apply some process discovery algorithms, which creates models, discover and visualise process that implement in the application, reflect the

behaviour that happens at runtime, not only that it is a way to infer workflows, but also capturing the behaviour in the log and to effectively present in a modern style.

Thirdly, the tool are expected to use one dashboard to connect the functionalities and features describe above, in order to make the analysing process more efficient and intuitive. The whole tool would brings the capability of extracting the data from BLF, and apply the process mining algorithm to it and present the result.

3.1.1 Extraction of Process Mining Data from Ethereum

Blockchain Logging Framework (BLF), which is a generic and highly configurable logging framework (Klinkmüller et al., 2020). Although process mining would do great help for analysing blockchain data, the raw data is not suitable for process mining, there is a mismatch between the logged data and the event data required for analysis, and extracting it can be highly inconvenient and very slow, also there are no logging libraries exists for Ethereum, furthermore, the source code for the Dapps may not be open to the public, which also causes the pain for decoding the encrypted information. So BLF for extracting process event data from Ethereum-based DApps created, allows users to query Ethereum in more flexible ways, retrieves data from Ethereum block by block in historical order, and extraction of data into formats suitable for analysis, such as CSV, TXT and XES files (Klinkmüller et al., 2019).

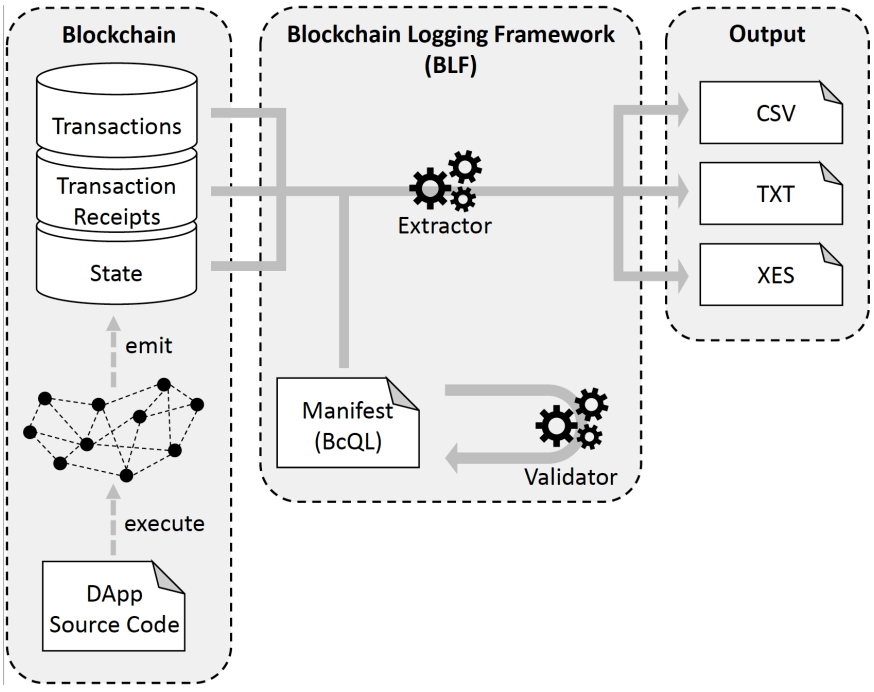

Figure 3. The components of BLF

As the Figure 3 shows, the components and the flow of BLF. There are several main parts, firstly, the Manifest is used to filtering which data is relevant, how data logged and how it should transform and formatted, allows user to specify their view of how data logged by the decentralised application should be interpreted. The Validator would follow the manifest rule, check and filter the data invalid format. Then Extractor applied with manifest and to iteratively retrieve data, transform the raw data into the target data format (Klinkmüller et al., 2020). Within all of those process, the final output would be extracted, as target files (XES, or CSV or other specified formats). Also, BLF contains several filters to make the process more flexible, for instance, block filters specified extracting a range of blocks, gives the access to the attributes in the blocks, transaction filter used to narrow down the transactions set based on address provides, the log entry filter allows user to choose log entry during the transaction, enables the user to access the attributes and event parameters, the smart contract state filter enables the user to query the smart contract's state stored, and the generic filter enables the user to filter data based on the specified criteria (Klinkmüller et al., 2020).

To sum up, BLF mainly focus on log extraction from Ethereum node into different data formats which support lots of use scenarios, minimising the size of demanded code (Klinkmüller et al., 2020). In this context, learning and applying BLF allowing us to continue analysing process and develop the tool of the project.

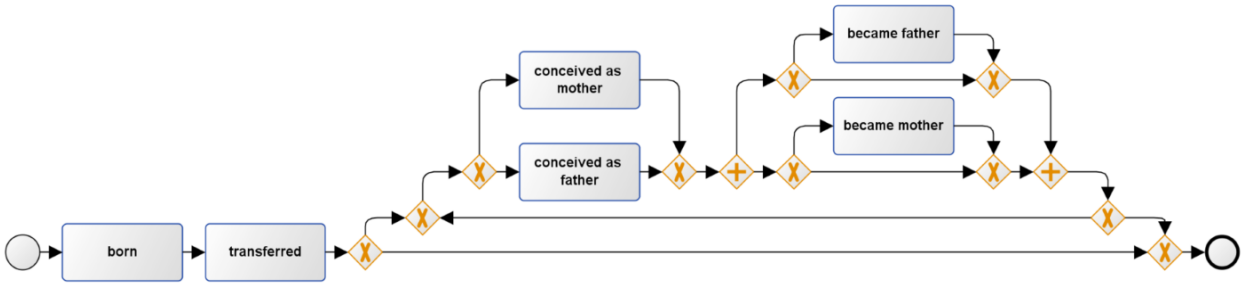

Figure 4. CryptoKitties lifecycle process (notation: BPMN) (Klinkmüller et al., 2020)

We look at a particular case as using BLF to analyse CryptoKitties. The methodology here used, firstly extracted logs from the smart contracts with BLF, each log entry has been mapped to individual events, and take the viewpoint of an individual kitty, by querying information related to the kitties' life cycles from log entries emitted. In this case, the lifecycle of each kitty is viewed as an independent process instance (Klinkmüller et al., 2019). Additionally, the block range was specified in first 3000 blocks for the log stems, those configurations would go in the manifest, and with the process of extracting, validating, and generating. After the whole process of BLF, the exported the data would be transformed and formatted into an XES file. After that, it is convenience for us to visualise the life cycle as a process model, as shown in Figure 4.

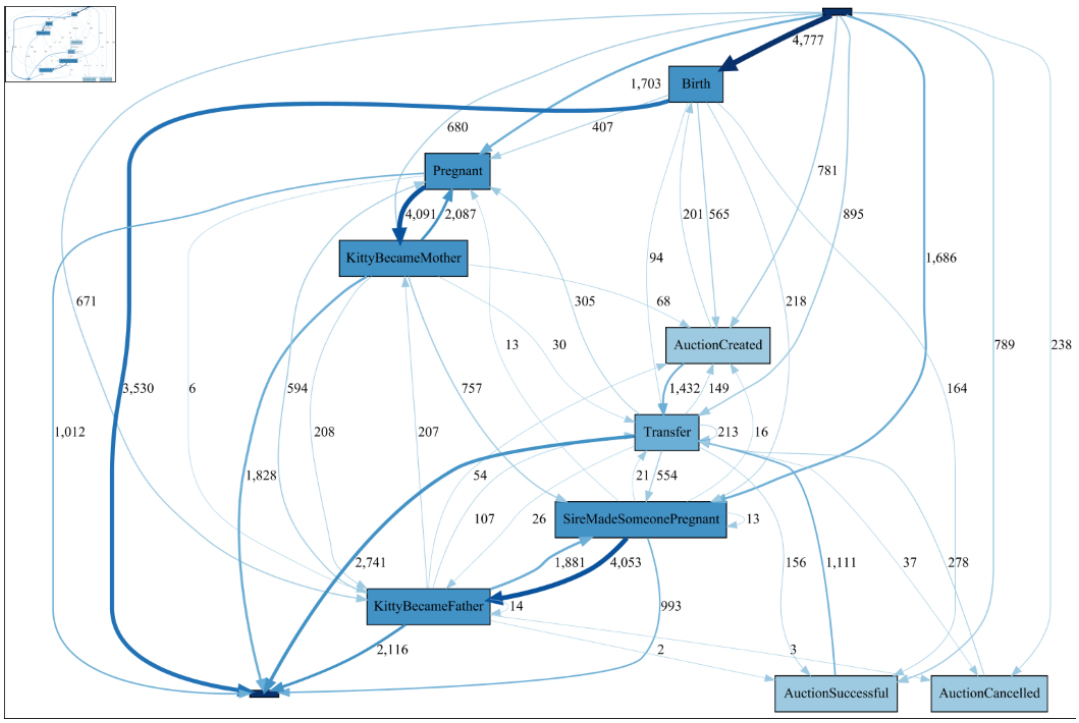

Figure 5. DFGs generated using ProM from logs

Moreover, the log extracted can also be load into proM, which is the extensible framework that supports a wide variety of process mining techniques in the form of plug-ins, and generate the process modal graph as Figure 5, which shows the kitty's behaviour every trace, including the variation and different kinds of event. Therefore, by applying BLF and analysing the output data using process mining algorithm, not only can dig into a deep analysis of kitty behaviour, but also confirms that the Dapp correctly implements the game rules, also shows the proposed framework can be applied to existing smart contracts (Klinkmüller et al., 2019).

There's similar work done by Mühlberger et al, in order to enable process analytics, the way they deal with non-trivial extraction and transformation tasks are different, but the purpose is also to convert raw data into an event log formatted such as XES standard. The algorithm they extract event logs data for process mining, is to generate function signature has and filtering transactions by comparing hash, by assigning transaction to process instance, to decode hash values, then after getting all matching transactional data, it can be transformed into XES format log (Mühlberger et al., 2019). As Figure 6 indicates the event log elements and blockchain raw data field.

Event log attribute	Blockchain data field
Case ID	Transaction.to
Activity ID	Transaction.input[2:10] (function selector)
Event ID	Transaction.hash
Activity label	Reverse-engineered from contract ABI and function selector in Transaction.input
Event timestamp	Block.timestamp (plus sorting by Transaction.index)
Event cost	Transaction.gas
Event resource	Transaction.from

```
{   blockHash:          0x1eca7ae74de59dff4f553b052a0e8346b1fc1587cf76ca5ee5b22da84f87822b,
    blockNumber:        1196772,
    from:               0x1387e74982055e3e1d235aad579350813b329b2b,
    gas:                1000000,
    gasPrice:           20000000000,
    hash:               0x656252f3ecee102d981520ca9e0ca0f7048bce99e4f6fead89d358cdbedd6156,
    input:              0xefe73dcb,
    nonce:              227,
    r:                  0xf26831a097ee1a3cf64364a07ff80fa816dd8604461482921a81be74276b5e7b,
    s:                  0x60a41b999ccd10461565d604cd063c299a5b1006a9ed8b0fcc84e5cfa9960f8d,
    to:                 0x0E6e0313dBe1Ba7A8bCb622EE7A77EaCBc9eF73f,
    transactionIndex:   3,
    v:                  28,
    value:              0 }
```

Figure 6. Mapping between event log elements and blockchain data fields

In this case, after the extraction process, extracts and transforms data stored on the blockchain to make it readily available for analysing using tools, they used the generated event log as an input for ProM to test its usage for process mining, the output is depicted in Figure 7, which illustrates the event log imported in ProM and visualised through the Dotted Chart plug-in, it can be seen the process instances were executed sequentially closely in the period of time (Mühlberger et al., 2019). Also, other plug-ins such as Inductive Visual Miner discovery and conformance checking plug-in can also be applied to turns out the result is confirmed as depicted in Figure 8.

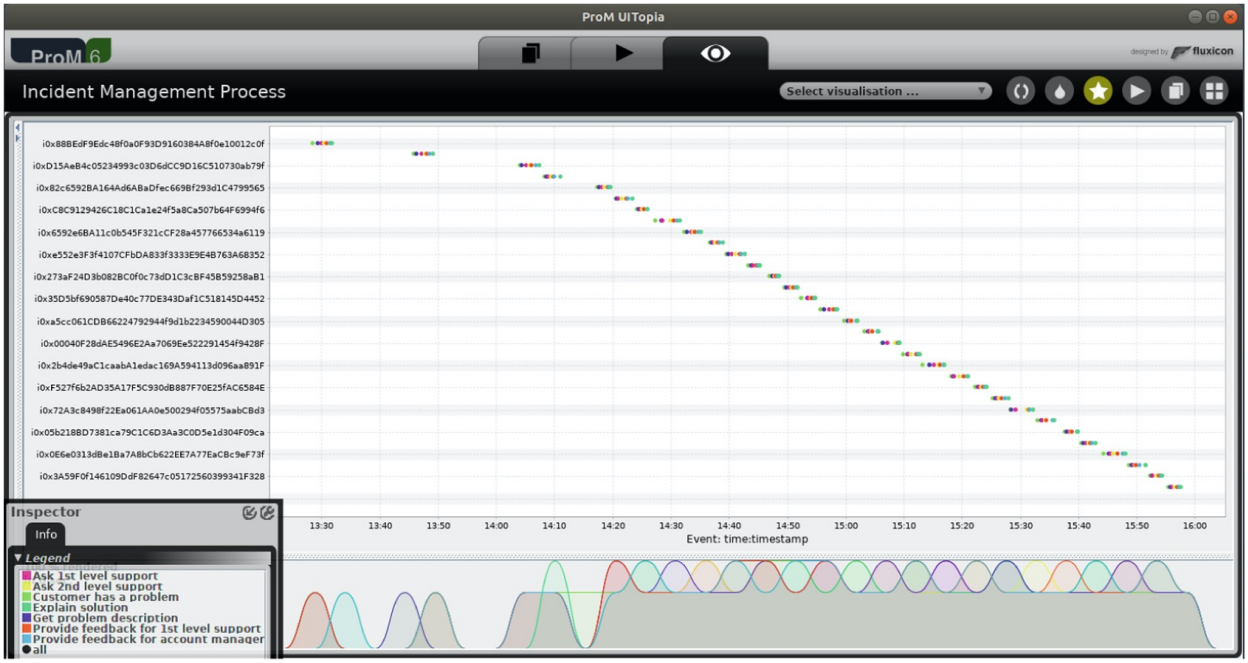

Figure 7. Dotted Chart visualisation of the incident management log in ProM.

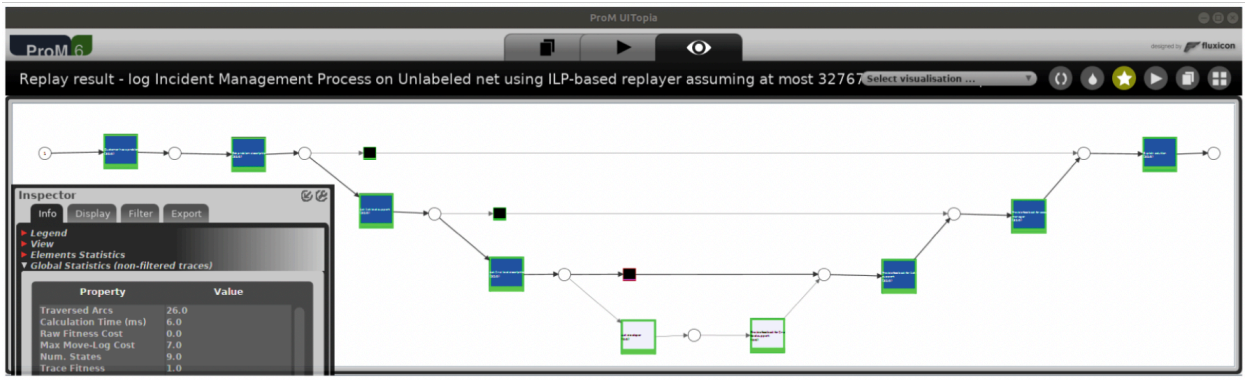

Figure 8. Conformance checking of the incident management process on the log

Hence, by analysing these aspects, indicates the model is 100% fitting with the recorded traces as expected, through this case, the presented outcome hints that the possibilities that the creation of event logs out of transaction data on the blockchain opens up, and has a wild range of use scenarios.

3.1.2 Graph Analysis of Blockchain Transactions

Chan and Olmsted performed their research aimed at using transaction visualisation to analysing the pseudo-anonymity in Ethereum, in other words, to verify if the anonymity holds true among the Ethereum platform, to find is there any possibility of tracing a certain address by analysis the transactions, and trying to find potential weaknesses of anonymity and recommendations for fixing

them (Chan and Olmsted, 2017), Since anonymity is a key promise of cryptocurrency platform. This case gives an example of the analysis of smart contracts, at the same time, well-illustrates the benefits of visualisation in Ethereum platform.

Figure 9. Graph Model of Neo4j

The first step for their methodology is to load the transaction data from REST API provided by Etherscan block explorer, which allows querying for all transactions to and from an address of an account, in doing so would reduce the time and disk required (Chan and Olmsted, 2017). It then loads the initial address from the transaction provided into the graph model of Neo4j, a graph analytics database, which the model can be seen in Figure 9. Then proceeds to load the adjacent addresses which are involved in one same transaction, and iterate over the transaction up to three iterations. In this process, also avoid the high throughput transaction node, in which these nodes might be tumbler service to gambling smart contact, and may cause the results even harder to interpret. After that, by applying Neo4j's graph query language, generates the graph depicted in Figure 10, divided accounts into blue and transactions into green, it could be clearly seen that transactions ending up in addresses that are associated with cryptocurrency exchanges based on Etherscan's tagging result (Chan and Olmsted, 2017).

They also mentioned another detail way of tagging, is to put the transaction data through a component called Clusterizer which tries to find groups of addresses that belong to the same user. After that, the next step is to create two graphs, transaction graph and user graph from the transaction data and output from Clusterizer. Then the third step is to enrich it with labels by the classifier of the resulting into a database. It provides a reverse index against the transactions, addresses, and users effectively (Chan and Olmsted, 2017).

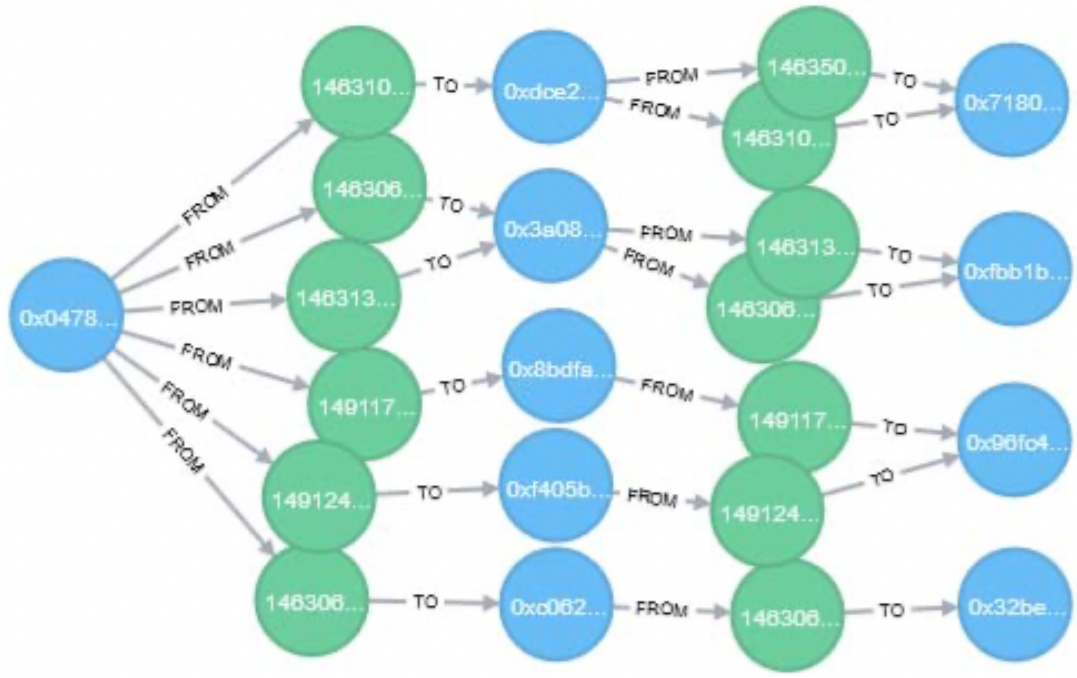

Figure 10. Transactions Associated Addresses Based on Tagging

Thus, in this use case, they focused on the process of how to visualisation the related transaction, by loading the whole Ethereum blockchain into Neo4j, generate a clear and neat way of manifestation. It can then leverage Neo4j extensions or other libraries to perform graph analytics, for example, the PageRank algorithm (Chan and Olmsted, 2017). The observation that the addresses associated with cryptocurrency exchanges tell that it is beneficial to link the transaction graph. Which this inspired us for the tools if we combine the extraction algorithm and the visualisation method, the understanding of the Ethereum platform and would have a better result.

3.2 Related Application of Blockchain Analysis Platform

In this section, we address two similar applications that have similar structures to the our proposed solution, or to some extent, shared their solutions that are inspiring for us.

3.2.1 BlockSci

The previous use cases we introduce the way of the extraction process in Ethereum and data visualisation, then here we introduce a software platform for blockchain analysis, called BlockSci. The structure depicted in Figure 11, there are two routes for importing data into BlockSci. One is

using the JSON-RPC interface as an importer, which is versatility since general blockchain platform mostly conforms to a standard JSON-RPC schema, another is their custom high-performance importer, which support reads directly from the raw blockchain data, and it supports the larger size of data, i.e. Bitcoin (Kalodner et al., 2017).

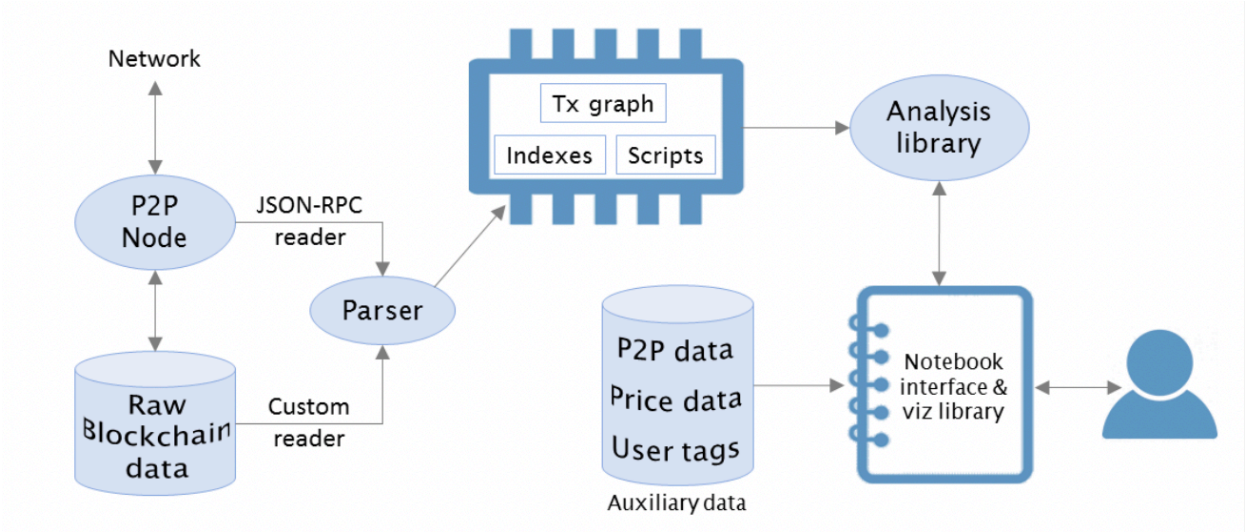

Figure 11. Overview of BlockSci's architecture

Through either route, the data is converted into the same format for parsing, rather than save the data on disk, which cost space and time, it directly passes into the parser, and converts into a common, compact format, in this process, every transaction input determines output it spends, encoded as transaction hash, it also maintains a mapping from the addresses to IDs for linking and prevent duplication. It also designs a filter for saving the space, since keeping the map all in memory would take too much memory, and keep it on disk would let the parser speed too slow, transactions and address stored in database on disk, list of the seen address stored in the cache, for optimising the address mapping, make a compromise between speed and memory. After that, the parser output is called Core Blockchain data, which is the primary dataset for analysis, contains transaction graph having the structure depicted in Table 1. The advantage of structure layout is spatial locality of reference (Kalodner et al., 2017). The analysing over transactions iteration shows benefit from caching, make the analyse process feasible even with insufficient memory to load the whole transaction graph, since the disk access is sequential.

Then the analysis library loads this data, which just parsed, as an in-memory database, allows the user can directly query or through a Jupyter notebook interface. (Kalodner et al., 2017)

The advantage of this tool is it has good performance, since it uses in-memory, analytical data base, and using algorithms to minimise the data, so it is highly efficiency and has good performance, the analysis process is faster. Also, it includes a library of useful analytic and visualization tools, such as

Description	Size
Size	32 bits
Locktime	32 bits
Input count	16 bits
Output count	16 bits
Outputs	128 bits each
Inputs	128 bits each

Table 1. Transaction structure

identifying special transaction. Moreover, The Jupyter notebook interface is suitable for exploration, and for high performance analyses requirement, BlockSci also has a C++ interface. These strong points, allows packaging together code, visualization, and documentation, makes it easy to share and reproducibility of scientific findings (Kalodner et al., 2017). However, it also has drawbacks, such as it does not support Smart contract platforms such as Ethereum, since its script is highly different from and more complex than other follow this basic structure such as Bitcoin's, we could focus on here, and draw inspiration from it.

3.2.2 Analysis of Business Process on Blockchain Infrastructure

This case shows investigate how to run a business process in terms of a supply chain on blockchain infrastructure, as to provide basic traceability of its run-time enactment (Di Ciccio et al., 2018). The integrating process such as supply chain usually need multi-party to work together, and usually comes with dense information exchange, the redundant data and the need of trust would always be issues for people to concern, therefore traceability of the good is necessary for either source tracking or responsibility assignment. Blockchain then would be a suitable solution to address these concerns, as its transparency and security, and the features that also make blockchain appropriate to execution of the business process along the supply chain. To enabling traceability of inter-organisational business processes, Di Ciccio et al perform the following methodologies:

- The first step of their approach is to transform the supply chain business process models into programs that are executable, they proposed using the Caterpillar tool, which is a blockchain-based business process management system. Figure 12 shows examples of addresses assigned by Caterpillar to the smart contracts, every process activity corresponding to a function of the worklist SC. Once a process instance is initiated through Caterpillar, a new block is added to the

chain each time an activity is executed. The problem with this approach is its process and task instances are based on Caterpillar, and the capability of handling BPMN pools in Caterpillar is under ongoing implementation (Di Ciccio et al., 2018).

	Global Factory SC address:	0x8cdaf0cd259887258bc13a92c0a6da92698644c0
	Process Factory SC address:	0x345ca3e014aaf5dca488057592ee47305d9b3e10
	Worklist Factory SC address:	0xf12b5dd4ead5f743c6baa640b0216200e89b60da
Prc. Instance	**Worklist SC address**	**Process Instance SC address**
1	0x6512a267ad28dfe41a5846e7ad0b2501633cb3f2	0xf2beae25b23f0ccdd234410354cb42d08ed54981
2	0x0ebe109b4ac5de65d63f7d7e5a856dcd77dc58fd	0xaa8f61728cb614f37a2fdb8b420c3c33134c7f69
3	0x22029e89e1d1f79d8e57c9af2fb9bf653bdf4be1	0xf21cf97429e6f7338ae989135ff9aa0225719347

Figure 12.Addresses of Smart Contracts deployed by Caterpillar on the blockchain.

- The second step is to integrate all the transactions from the corresponding worklist SC to trace the execution of a process instance, then search for matches between the value of the contract address field in the transaction and the worklist SC in Caterpillar as shown in Figure 13, which illustrates the structure for traceability. Therefore, after comparing the contract address of a transaction to the hash-code of the worklist SC to identify the process instance (Di Ciccio et al., 2018).
- Then, match the function identifier of the transaction with the hash-code of function signatures to retrieve the activity.

To Sum up, it mainly identifies and links the transactions of running process actives, presents a technical solution to ensure traceability through the blockchain business process (Di Ciccio et al., 2018), shows correctness and conformance of the approach.

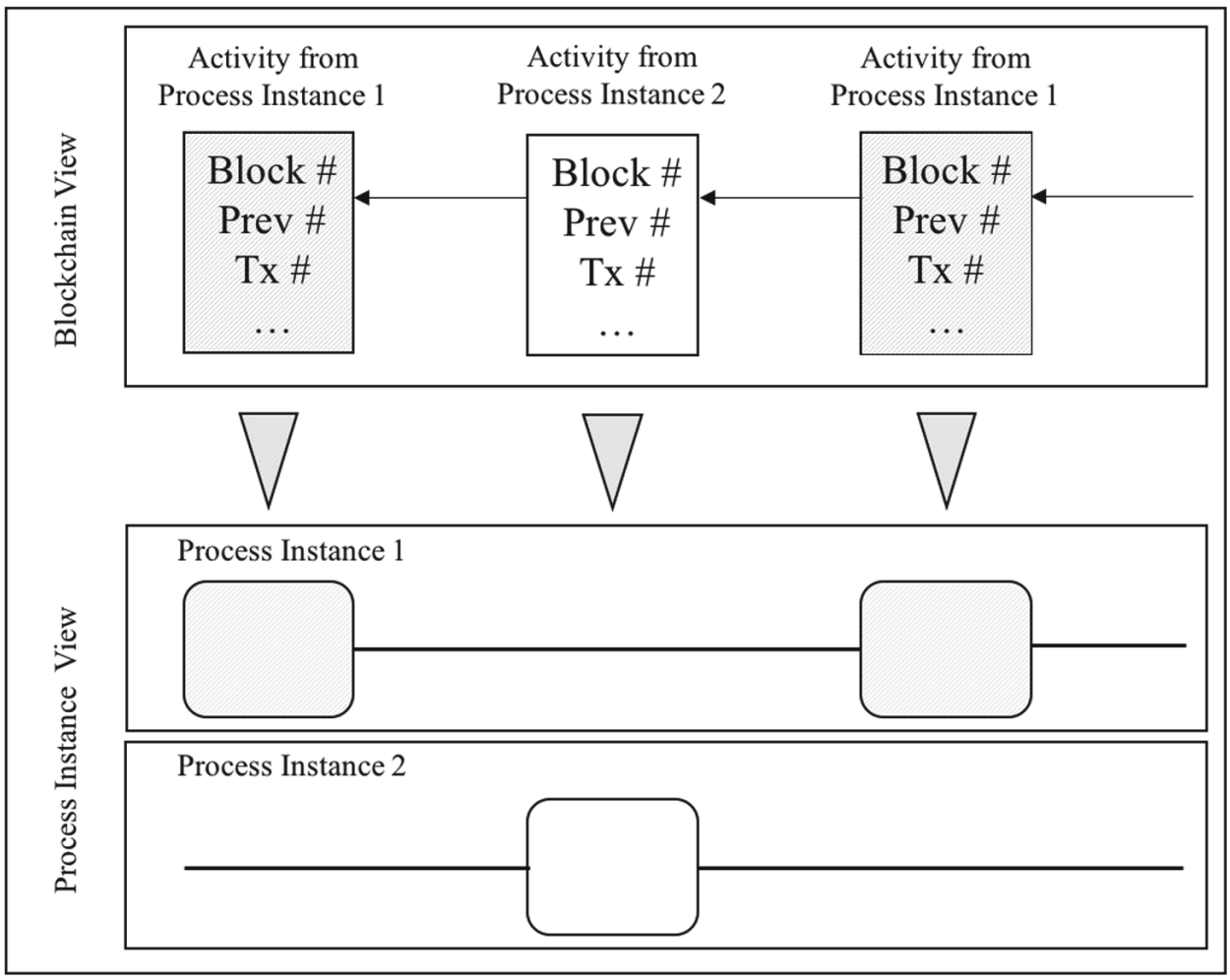

Figure 13. Extraction of process instances

The results of the methodology illustrate the applicability of the automated techniques to gives traceability of processes deployed on the blockchain. However, Di Ciccio et al also acknowledge the drawbacks, foremost is the identification process and comparison relies on Caterpillar, and the capability of Caterpillar still needs further development. Also, the approach to identify processes and instances requires all the parties to share knowledge on the hash key and hashing function in use (Di Ciccio et al., 2018), which may need further work and investigation.

The results collected from the work via their implemented prototype show the possibility and accuracy of the methodology thus makes a contribution towards a comprehensive framework.

Chapter 4

Tool Design

4.1 Software Design Overview

For the entire analysis process, the steps to put extracted blockchain data into use:

- Firstly, extract the data, users need to specify the details such as blocks range or smart contract address to extract the relevant data from the Ethereum network.
- Secondly, process the data, transform and filtering the useful information from the raw event data extracted into the data format that is suitable for subsequent analysis.
- Thirdly, discover the model, applying the process discovery algorithm to turn the data recorded into knowledge of an application's business processes. (Eck et al., 2015)
- Finally, inspect the model to obtain insights.

However, the literature review in Section 3 revealed a few limitations of existing work:

- BlockSci (Kalodner et al., 2017) used do not support smart contract platforms, and it does not provide the capabilities of visualization.
- BLF mainly focuses on extraction but need to combine with other analysing tools.
- Neo4j (Chan and Olmsted, 2017) used does not involved with the extraction process, for this they rely on a third-party Etherscan.
- Caterpillar (Di Ciccio et al., 2018) used, its capability of handling BPMN pools in Caterpillar requires further implementation, and it mainly focuses on traceability only.

So that, there's no perfect tool to cover and connect the whole analysis process, the existing tools only address individual parts of the process. Hence our tool is designed to connect the tools, reuse the ideas from existing work. In order to achieve this, we designed the tool is a web-based application since the navigation and page-style layout is good for and familiar to the users, and all of the functions can be integrated based on this.

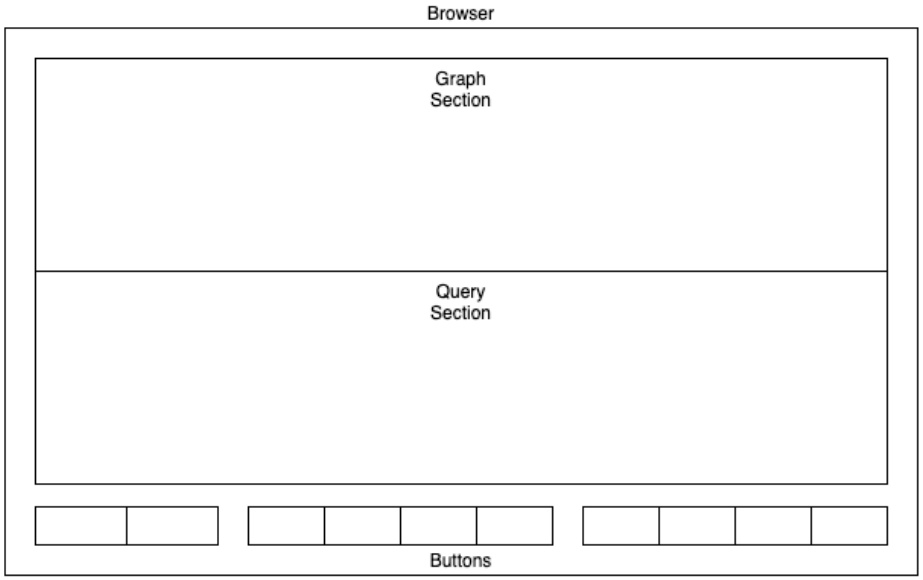

Diagram 1. UI Design

The UI design for the project is shown in Diagram 1, which supports the analysis process. The "Query Section" is for extracting purpose, in order to put query here and users are able to adjust the query through this section, hence adjust the criteria for the extracting process. Users are also had options to download and upload the query for future use. The "Buttons" section is in charge of handling the user's request to perform actions, such as extracting and transforming the data, the start/ stop of each process can also be controlled via this section. The "Graph Section" is for discover and inspect the process model that generated by the process discovery algorithm, once the DFG is finalised, it automatically presents in this section and users are able to interact with it, users are also have options to either download the DFG as an image, or download the log file just transformed for future use.

The reason to choose DFG among other process discovery algorithm is due to its simplicity. Because of this, DFG is generated faster than other algorithms, such as InductiveMiner, and is more efficient for small to median data sets. But the compromise is that there are some flow details that DFG cannot show, such as average execution time, but considering the development time and complexity of the code, DFG is chosen as the final process discovery algorithm, more detail is addressed in the next section 4.2

Thus, the entire analysis process list above is covered by this design, which supports the overall methodology.

4.2 Iterations

As the development went on, we were refining software deign. According to the specific feasibility of a scheme, the development time, as well as the complexity of the implementation, our tools went through many iterations. The final version was proposed in Section 4.1, and in this section we just elaborate on each version change and rational for each decision we've made.

At first, tools should integrate some existing tools into the dashboard, because these tools usually integrates enough process mining algorithms and models to support our analysis, whether it is process discovery, conformance checking or model enhancement, so integrating it into our tools has become our first choice. However, due to code limitations, developing the dashboard inevitably make it very difficult to integrate another non-Web-based tool into it. And the time cost and difficulty of trying to do so is far greater than writing a corresponding process discovery algorithm by ourselves. At the same time, the progress and development of the web dashboard had been carried out for a while, so it was obviously not cost-effective to give up the existing progress and re-plan and re-develop. Even though its features were very abundant, we still chose to give up the idea and change our original plan.

Our second updated plan is to realise the visualization of one model by selecting a simple process mining algorithm, and then add other models one by one. We ended up with Directed-Follows-Graph, because of its simplicity as mentioned is the last section, it is not difficult to develop this algorithm on the basis of the web-application, so we chose this algorithm for our second version tool. Moreover, DFG mining is commonly applied and easy to understand for nontechnical people. An alternative to DFG mining would be to use advanced process discovery algorithms which however require more skill.

And then is the environment iteration. In the early stages of development, we've tried using lots of the ways to set up a local private network on the local laptop, the benefit for these choices are the data access speed is faster since everything is running locally, but the limitation is that it may not adapted with some of the tools, which cause the failure and conflict of them. Another option is to use a public Ethereum network, the benefit of it is that it contains enough data and most of the tools is adapted, but the drawbacks are it's low speed of data access, for each extracting process takes way more time than the local network. The final decision is to use the public ethereum network due to its good adaptability, since our goal is to integrate the tools together, so that's the priority to consider about.

Chapter 5

Implementation

Now we've addressed the software design and related works, in this chapter, we present the implementation detail that contain: Software Architecture (4.1) and Techniques (3.2) which including Blockchain Logging Framework (3.2.1), Process Discovery Algorithm (3.2.2), and a Graph Visualisation Layout Algorithm (3.2.3).

5.1 Software Architecture

So the software architecture for the tool is a web dashboard, the whole dashboard is designed as a server-client webpage as depicted in Diagram 2. The frontend user accesses the dashboard, defines and executes queries and then sends requests and information to the backend, which calls the

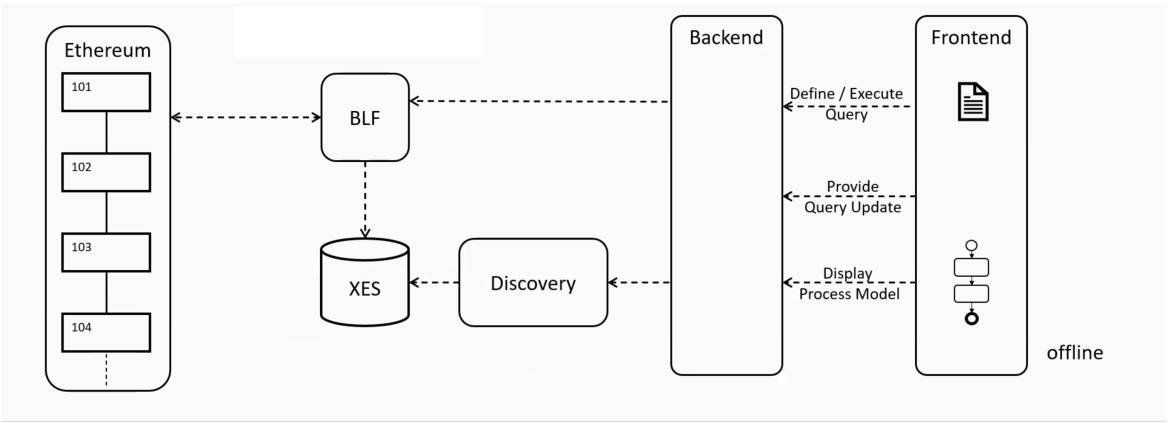

Diagram 2. Software Architecture

Blockchain Logging Framework component to extract data from the Ethereum network, and when the data is finalised, The BLF is then sent it back and formatted as an XES file, which is can either downloaded or straight into the process discovery algorithm, which is the directed following graph, and then through the back-end and processing results, it eventually presents to the front-end user. This is the complete workflow.

As for the role of the components, the dashboard would be responsible for the routing and navigation of the website, handling clicking event and interface interaction, also handling the data passing into the backend, it calls the API backend and deliver information to its corresponding parts to render the user interface. The HTML and CSS would be responsible for the styling and appearance of the dashboard. In the backend, BLF would be used for handling requests of extracting data process, after that, the target files would be passed into the process discovery algorithm, in order to accomplish analysing using process mining algorithms, then React JS would re-render the dashboard to show the result.

Firstly for the data extraction part, we have proposed that users are able to call the components from BLF via the tool, they are able to compose the query at the dashboard, which is a language used by BLF to define the criteria and configure the parameters, meanwhile user can also upload and save the query they've composed for future use. User can also validate the query through the Validator from BLF, because it is needed to follows the manifest rule before actually running BLF to extract, what validator does is to check the user-composed query is in a valid format. And most importantly user should be able to execute the query and actually running BLF to perform the data extraction function, the result is exported to the target format log file for further use.

Secondly for process discovery algorithm part, we proposed one algorithm called Directed-Follows-Graph (DFG) for the given data-set, this algorithms take each trace and events from the input, and export a visualised graph get back to the users, so this process discovery algorithm would turn a raw log data into a modern graph that indicates the process based on the events.

Thirdly from a higher level, the tool should broken down the barrier between the tools, once BLF is finished, the extracted data can be either downloaded, or straight apply with the Directed-Follows-Graph, after processing and dealing with the data, the graph is automatically present to the user.
And all of these functions are integrate into a interactive and responsive dashboard, to complete the flow of the analysis process

5.2 Techniques

In this section, we address how the actual techniques are implemented including BLF, DFGs, Graph Layout Algorithm and Web-Based Integration.

5.2.1 Blockchain Logging Framework

The first feature is event log extraction, which allows users to pull data from the Ethereum network for subsequent analysis. The technology we used here was BLF (Blockchain Logging Framework).

The entire BLF extraction step is made up of several parts, the extraction/transformation and validation is all supported by BLF. First of all, it has a query language that allows the user to implement the entire function using only this query language. And it turns the user compose query code into the BLF built-in filters, such as block filters specified extracting range of blocks, transaction filter used to narrow down the set of transactions based on address, and etc, so that to narrow the range of data, reduce time, increase efficiency, and improve data correlation. Secondly, when The user validates the query, it calls The BLF validator, In which would follow the manifest rule, check and filter the data invalid format. And for the actual execution parts, It calls BLF extractor applied with manifest and to iteratively retrieve data, extracts event data from Ethereum block-by-block via a WebSocket connection, transform the raw data into the XES file and export. Therefore, during the whole process, BLF minimising the amount of required code, as long as the user know the query language and its usage, which improves the use efficiency of the whole tool, and It also provides data support for the subsequent process discovery algorithm.

Another technology that we've used is different kind of Ethereum network, so currently we are using Infura as the public Ethereum network provider, as the web socket connection is perfectly adapted with BLF, we've also tried different kinds of local private networks, configure the best one that works for the tool, but this brings the biggest problem that takes me almost one semester to figure out.
During the last semester, we've tried using lots of the ways to set up a local private network on my laptop, but those are not adapted with BLF, so we finally use public Ethereum networks instead of them, more rational of decision is addressed in chapter 7.2.1

5.2.2 Process Discovery Algorithm

The second feature is Process discovery, once the XES data is exported from BLF and it's applied with process discovery to visualise the model of application, the functionality for these part is firstly — visualisation, by turning the data file into a modern, clear graph, so that it is convenient for users to analyse specific cases. Second function here is to transform the data, what we did here is read the XES file content, and classify each trace, and for each single trace, identify the name of events and the relationship to others, and once you have these data graph can be easily draw with some external packages.

We have use the Direct-follows-graph implemented ourselves as the process discovery algorithm, which would take the data just transformed, treat each unique event as a node and treat each event-to-event relationship as a link, and then iterate through the entire log data, after the transforming, the data-set is ready to input to an web-based graph library called D3-react, which is based on D3, and is a Javascript library for rendering graph to the dashboard using HTML, SVG, and CSS.

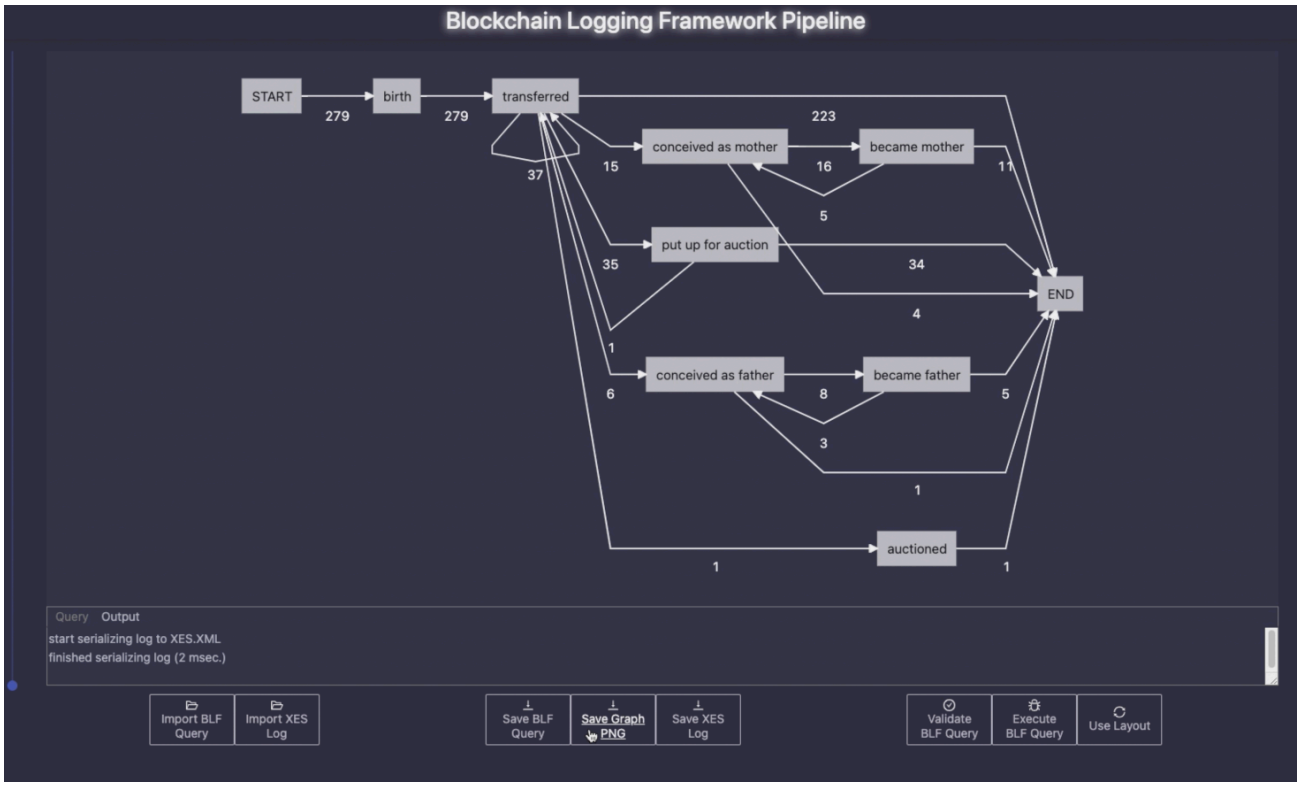

Diagram 3. Used Layout Graph

5.2.3 Graph Visualisation Layout Algorithm

The layout algorithm also used for the Direct-follows-graph, in order to make the graph more formal, we used Dagre — which is a JavaScript library that makes it easy to lay out directed graphs on the client-side. So that by using Direct-follows graph, we can convert an extracted event data into human understand-able process descriptions and visualised it as depicted in Diagram 3, which further helps user to monitor the process of application and understanding the user behaviour.

Also, another mode provided, which allows user to interact with the graph, without any layout algorithms, users are able to drag each single node of the graph, at the same time, the relationship between each node is revealed as depicted in Diagram 4, hence enhance the usability of the tools and increase the flexibility of the graph.

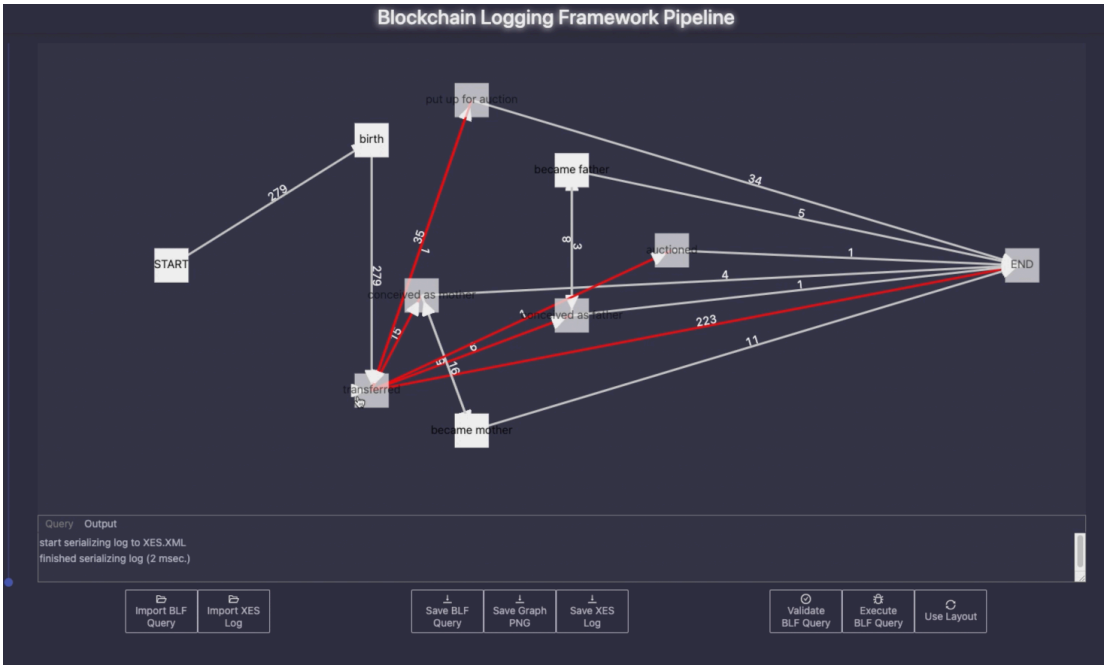

Diagram 3. Interactive Graph Mode

5.2.4 Web-Based Integration

For integration, we takes time to put all of the function together and fit into one dashboard, user can type the query in, push to the Blockchain login framework, query the data, and the execution result would import into the process discovery algorithm, and visualised in the backend. In this way, all the functions are integrated and we have a complete tool. We've also focused on the

usability improvement, including the interface styles, colors, components layout, or the way user interactive with each functions, we also take some effort to make the tool as usable as possible.

The technology used here, for frontend we use React Javascript with HTML and CSS, since it has abundant external libraries for extensional functions, which contribute to the dashboard implantation, allows user to interact with. For the backend, we use NodeJS with Express, to handle request that user send, and dealing with the execution and other processing tasks.

Chapter 6

Evaluation

This chapter includes Result (6.1) shows the details have done so far. Also Discussion (6.2) have made based on the outcomes and process, including the problems encountered and challenges (6.2.1).

6.1 Results

So in summary, since it integrate lots of functions to make the analysis process together, and now the entire process is less fragmented, and it's more easier to do process mining on blockchain data since users don't need to go and manually perform each single functions one by one, for example, users don't need to connect the network first, use tools to extract the data, and then use the data into another tool to perform the analysis functions, and now the whole process is well-connected and straight forward to the user, they only need to specify essential data, the result is happening automatically.

Table 2. is the completeness table of the planned tasks, as the task shows, we integrated each components and functions and connect the whole process, also for the usability improvement, we've focus on the UI&UX design and to have a responsive interactive interface, moreover, Directed-Followed-Graph also successfully implanted by using external javascript libraries. Furthermore, additional functions provides to the user, enables more options such as saving and downloading for the XES log and Graph. As for conformance checking, this part was out of scope for thesis C and we may leave it for further work.

Planned tasks	Completeness
Integration of each part	Connecting each components
Usability improvement	Responsive interactive interface / UI improvement
DFG algorithm	Dagre D3 package
Additional functionality	Query saving/XES uploading/ result downloading
Conformance checking	Out of scope for thesis c

Table 2. Plan Completeness Table

7.2 Discussion

7.2.1 Challenges

And then we encountered another problem, which is the biggest problem for the whole project. In the early stages of development, we've tried using lots of the ways to set up a local private network on the local laptop, for example GETH and Ganache for local Ethereum environment, and Truffle for smart contact deployment and interaction, but somehow BLF cannot extract data from these Ethereum network, which directly led to the fact that in that versions, BLF could not be integrated into the tool because of the issue of private network. Since BLF was responsible for data extraction and transformation, which were essential to the entire project, we urgently used another alternative to perform similar work which is Web3JS, which is an external Blockchain library.

So in that version, you need web3JS to access ethereum data, but this method has its own deadly features. Web3JS does a good job adapting front-end code such as Javascript, but the data obtained by Web3JS is not automatically converted properly. What it returns is just a string of JSON text, which needs additional data processing and transformation to meet the requirements of process mining. Also, the amount of code required is much greater than the amount of code needed to integrate BLF, moreover, the amount of work needed to extract Ethereum event data with web3JS is way more than BLF, in addition to the smart contract addresses and signatures, block ranges required for BLF queries, users also need to know the smart contract ABI and

transaction address, which requires a lot of extra code and computation time, largely reduce the efficiency and increase the cost.

Fortunately, the problem of BLF with private networks were finally solved at the beginning of Thesis C, which is to use an existing public Ethereum network Infura with WebSocket connections, which is, then the bad-choice of web3JS can be replaced with BLF.

After that, there were several minor design changes, such as replacing the filling-in filter input mode with the whole BLF query compose section, or updating the layout algorithm of DFG, etc. However, these are not major updates, and more implementation details are address in chapter 5.2.

During the development, we do encountered some problems and constrains, such as for the layout algorithms Dagre, it can only shows node-to-node relationships, but for each link, it cannot adjust the length of the edge based on the edge weight. Another problem is the efficiency of the graphics library D3.js in dealing with large data-set, which can take a long time, and add the time spent on transforming the event data, which would make the whole process very lengthy, in order to solve this, we used the library which is currently using called D3-React (instead of the native D3.js), this is more adapt with the React JS language and the speed for drawing small-medium sized graphs is quicker than it used to be.

Also, the problem have encountered for integration is when user send the request, some of the request may not be accomplished in order since the execution function is asynchronous, so for example when blockchain logging framework is not finish extracting the data, but the directed-follows-graph would automatically read the empty file, so this may cause the error, and this problem is resolved by using another structure of handling request, such as using a different javascript hooks to avoid unordered request or execution.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

Throughout this thesis, we have examined the use scenario of the blockchain platforms, particularly Ethereum, also dig into the details of the benefits of analysing blockchain data, and show the capability of the analytics tool to solve the existing complications. Also, introduced the related works and investigations, evaluates their contributions and drawbacks.

We proposed, designed and developed the tool that aim to use process mining for blockchain data analytics, we believe we are moving in the right direction, and elaborate on the design ideas, software architecture and the code implementation.

In moving forward, we will try to improve the actual tools with further functions, and potentially attempt to improve and put our effort into a far wider range of use scenarios, that help address the issues people are experiencing.

7.2 Future Work

At moment, we can only do Directed-Follows-Graph, but in the future, that might not be enough for process mining, we might need more mature discovery algorithms. Similarly, in order to enable users to do more analysis , we might need more algorithm models, such as BPMN, Business Process Model and Notation, since at the moment, DFG only shows the control flow, but it’s not possible to dig into the details such as certain behaviours under certain circumstance, such as the average execution time between different activities, and etc. The analytical scope of the tool is limited, and that’s the reason to extended the tool’s core functionality in the future.

Another point that can be further improved is the efficiency of the tools. Due to the drawbacks of the DFG layout algorithms, as well as the BLF extracting time, as you saw in the demo, the whole process would take long time when dealing with large data set, for some case even take hours/days to output the result, which is time consuming and not efficient, that's why obviously the performance of the tool, the entire mining process need to be speed up, and That's another point can be further focus in the future.

Bibliography

1. Chan, W. and Olmsted, A., 2017. Ethereum Transaction Graph Analysis.

2. C-sharpcorner.com. 2021. Setup Your Private Ethereum Network With Geth. [online] Available at: <https://www.c-sharpcorner.com/article/setup-your-private- ethereum-network-with-geth2/> [Accessed 29 April 2021].

3. danielcaldas.github.io.2021. react-d3-graph2.6.0|Documentation.[online]Available at:<https://danielcaldas.github.io/react-d3-graph/docs/index.html>[Accessed29 April 2021].

4. Di Ciccio, C., Cecconi, A., Mendling, J., Felix, D., Haas, D., Lilek, D., Riel, F., Rumpl, A. and Uhlig, P., 2018. Blockchain-Based Traceability Of Inter-Organisational Business Processes.

5. Eck, van, ML, Lu, X, Leemans, SJJ & Aalst, van der, WMP 2015, PM2 : a Process Mining Project Methodology. in J Zdravkovic, M Kirikova & P Johannesson (eds), Advanced Information Systems Engineering : 27th International Conference, CAiSE 2015, Stockholm, Sweden, June 8-12, 2015. Proceedings. Lecture Notes in Computer Science, vol. 9097, Springer, Berlin, pp. 297-313, 27th International Conference on Advanced Information Systems Engineering (CAiSE 2015), Stockholm, Sweden, 8/06/15. https://doi.org/10.1007/978-3-319-19069-3_19

6. Fleder, M., Kester, M. and Pillai, S., 2015. Bitcoin Transaction Graph Analysis.

7. Geth.ethereum.org.2021.Command-lineOptions|GoEthereum.[online]Available at:<https://geth.ethereum.org/docs/interface/command-line-options>[Accessed29 April 2021].

8. GitHub.2021.GitHub-dagrejs/dagre-d3:[DEPRECATED]-A D3-based renderer for Dagre. [online]Available at:<https://github.com/ dagrejs/dagre-d3> [Accessed 11 July 2021].

9. GitHub.2021.TU-ADSP/Blockchain-Logging-Framework.[online]Available at:<https://github.com/TU-ADSP/Blockchain-Logging-Framework>[Accessed29April2021].

10. hackernoon.com. 2021. Ethereum Development Walk through (Part2: Truffle, Ganache, Geth and Mist)|HackerNoon.[online] Available at:<https://hackernoon.com/ ethereum-development-walkthrough-part-2-truffle-ganache-geth-and-mist-8d6320e12269> [Accessed 29 April 2021].

11. Hobeck, R.,Klinkmüller, C., Bandara, D., Weber, I. and van der Aalst, W., 2021. Process Mining on Blockchain Data: A Case Study of Augur.

12. HUANG, Y., WANG, H., WU, L., TYSON, G., LUO, X., ZHANG, R., LIU, X., HUANG, G. and JIANG, X., 2020. Understanding (Mis)Behavior On The EOSIO Blockchain.

13. Kalodner, H., Goldfeder, S., Chator, A., Möser, M. and Narayanan, A., 2017. Blocksci: Design And Applications Of A Blockchain Analysis Platform.

14. Kell, T.,Yousaf, H.,Allen,S.,Meiklejohn, S. and Juels, A.,2021.Forsage: Anatomy of a Smart-Contract Pyramid Scheme.[online]arXiv.org. Available at: <https://arxiv.org/abs/2105.04380> [Accessed 24 July 2021].

15. Klinkmüller, C., Ponomarev, A., Tran, A., Weber, I. and van der Aalst, W., 2019. Mining Blockchain Processes: Extracting Process Mining Data From Blockchain Applications.

16. Klinkmüller, C., Weber, I., Ponomarev, A., Tran, A. and van der Aalst, W., 2020. Efficient Logging For Blockchain Applications.

17. Mühlberger, R., Bachhofner, S., Di Ciccio, C., García-Bañuelos, L. and López-Pintado, O., 2019. Extracting Event Logs For Process Mining From Data Stored On The Blockchain.

18. Müller, M. and Ruppel, P., 2019. Process Mining For Decentralized Applications.

19. M.P. van der Aalst,W., 2019. A practitioner's guide to process mining: Limitations of the directly-follows graph. Fraunhofer Institute for Applied Information Technology, Sankt Augustin, Germany.

20. Nakamoto, S., 2008. Bitcoin: A Peer-To-Peer Electronic Cash System.

21. Paquet-Clouston, M., Haslhofer, B. and Dupont, B., 2019. Ransomware Payments In The Bitcoin Ecosystem.

22. promtools.org. 2021.start| ProM Tools.[online]Available at:<http://www.promtools.org/doku.php>[Accessed29April2021].

23. Remix.ethereum.org. 2021. Remix - Ethereum IDE. [online] Available at: <https://remix.ethereum.org/> [Accessed 29 April 2021].

24. Ron, D. and Shamir, A., 2012. Quantitative Analysis Of The Full Bitcoin Transaction Graph.

25. Truffle Suite. 2021. Ganache | Creating Workspaces | Documentation | Truffle Suite. [online] Available at: <https://www.trufflesuite.com/docs/ganache/workspaces/ creating-workspaces> [Accessed 29 April 2021].

26. Truffle Suite. 2021. Truffle | Interacting with Your Contracts | Documentation | Truffle Suite. [online] Available at: <https://www.trufflesuite.com/docs/truffle/getting- started/interacting-with-your-contracts> [Accessed 29 April 2021].

27. Van der Aalst, W., 2016. Process Mining. 2nd ed. Springer Berlin Heidelberg, pp.1-52.