Process Mining on Blockchain Data: A Case Study of Augur

Richard Hobeck[1(✉)], Christopher Klinkmüller[2], H. M. N. Dilum Bandara[2], Ingo Weber[1], and Wil M. P. van der Aalst[3]

[1] Chair of Software and Business Engineering, Technische Universitaet Berlin, Berlin, Germany
{richard.hobeck,ingo.weber}@tu-berlin.de
[2] Data61, CSIRO, Sydney, Australia
{christopher.klinkmueller,dilum.bandara}@data61.csiro.au
[3] RWTH Aachen University, Aachen, Germany
wvdaalst@pads.rwth-aachen.de

Abstract. Through its smart contract capabilities, blockchain has become a technology for automating cross-organizational processes on a neutral platform. Process mining has emerged as a popular toolbox for understanding processes and how they are executed in practice. While researchers have recently created techniques for the challenging task of extracting authoritative data from blockchains to facilitate the analysis of blockchain applications using process mining, as yet there has been no clear evaluation of the usefulness of process mining on blockchain data. With this paper, we close that gap with an in-depth case study of process mining on the popular Ethereum application *Augur*, a prediction and betting marketplace. We were able to generate value-adding insights for application-redesign and security analysis, as validated by the application's chief architect and revealed blind spots in Augur's white paper.

Keywords: Blockchain · Process mining · Case study · Process discovery · Conformance checking · Ethereum

1 Introduction

A blockchain can be characterized as a distributed, append-only data store for transactions [31]. Second-generation blockchains have comprehensive smart contract capabilities, i.e., allow for the deployment and execution of user-defined programs. On this basis, blockchain has emerged as a technology allowing the automation of cross-organizational processes on a neutral platform [20,29].

Process mining [1] has become popular as a toolbox for understanding processes and how they are executed in practice. For example, many case studies ranging from healthcare [5,19,25,27], finance [8,12], manufacturing [26], and public services [3,17] to software development [18] applied process mining to analyze processes from different perspectives including aspects, such as

A. Polyvyanyy et al. (Eds.): BPM 2021, LNCS 12875, pp. 306–323, 2021.
https://doi.org/10.1007/978-3-030-85469-0_20

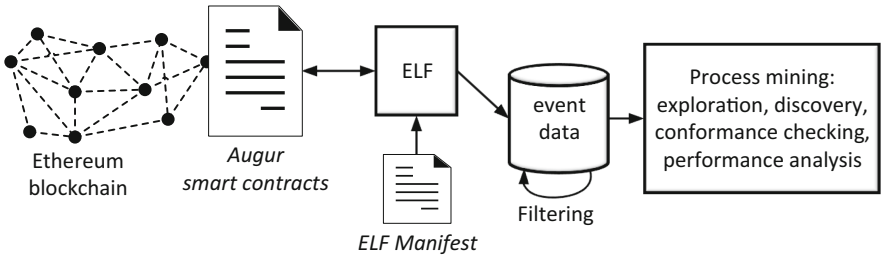

Fig. 1. Overview of the approach.

control flow, conformance, drifts, and performance [24]. Nevertheless, process mining on blockchain data turned out to be a challenging task [10]. Hence, recently researchers have created techniques to extract authoritative data from blockchains [13,14]. On that basis, concepts were introduced that can put extracted blockchain data into use, e.g., for monitoring business processes executed on a blockchain [9]; validating smart contracts on Hyperledger Fabric [11]; auditing blockchain applications on Ethereum [7]; analyzing transactions stored on the Ethereum network without focusing on specific *decentralized applications* (DApps) [21]; and using process mining on blockchain-wide data [22] but no single process in particular. As yet, there has been no clear evaluation of the usefulness of process mining on blockchain data. All of the above process-specific approaches have been evaluated with small examples, demonstrating technical feasibility more than the usefulness and business value.

With this paper, we close that gap and analyze the usefulness of process mining on blockchain data with an in-depth case study of process mining on the popular Ethereum application *Augur*[1]. Augur is a prediction and betting marketplace, where users can create bets (e.g., "Will Donald Trump win the 2020 U.S. presidential election?"), and other users can bet on the outcomes. Because Augur smart contracts run on the public Ethereum blockchain, all data are time-stamped, transparent, and available. We used our *Ethereum Logging Framework* (ELF) [13,14] to extract Augur data. This extraction resulted in nearly 3000 traces and more than 23000 events. As shown in Fig. 1, we then filtered the data and applied various process mining techniques to analyze Augur from control flow, conformance, and performance perspectives.

In our study, we were able to generate insights of value to the business. In more detail, we provide a clear view of how Augur is used, verify its design mechanisms, and check for unintended behavior and bugs in the (immutable) code; immutability poses a challenge from a *business process management* (BPM) perspective [20], and software engineering in general [28]. The usefulness of the insights was confirmed by anecdotal evidence of Augur's chief architect, particularly in terms of understanding user behavior and code validation, which is

[1] https://augur.net/, accessed 2021-03-05.

especially relevant for security aspects in an open-source application that can be invoked (and thus potentially attacked) by anyone with Internet access.

The paper is structured as follows. Next, we introduce the object of our case study, Augur. Then, Sect. 3 outlines the data extraction and pre-processing procedures. The focal point of the paper is the data analysis in Sect. 4, covering data exploration, process discovery, conformance checking, and performance analysis. Finally, we discuss the results in Sect. 5 and conclude in Sect. 6.

2 The Case of Augur

Augur is a betting platform and prediction marketplace that is implemented as a set of smart contracts on the public Ethereum blockchain. Augur's white paper characterizes the mechanics of a prediction and betting market: "individuals can speculate on the outcomes of future events; those who forecast the outcome correctly win money, and those who forecast incorrectly lose money." As a betting market organized on Ethereum, the developers claim that Augur bypasses disadvantages of traditional betting markets, such as trusted market operator and limited participation [23].

Currently, there are two versions of Augur available in parallel: Augur v1.0 (launched 2018-07-09) and Augur v2.0 (details announced April 2020[2], launched 2020-07-28[3]). Interestingly, to gain user trust, the Augur developers open-sourced the smart contracts and deployed both versions without any option to update or stop them – giving themselves the privilege to do either might result in the loss of users' cryptocurrency and omitting such a possibility; therefore, increases trustworthiness. Hence, the new version is deployed in parallel to the old one, as such *not comprising an update* in any traditional sense. However, once the new version was deployed and users migrated to it, the old version became "economically insecure" according to the developer team, and therefore should not be used anymore. Because prediction markets are long-running, and hence extended observation time frames are crucial for their analysis, we nevertheless focused on Augur v1.0 and considered the data from its launch until its use was no longer recommended in July 2020 (see Sect. 3 for details).

Augur was chosen for this case study for several reasons. *Data availability.* Augur v1 was among the most popular Ethereum DApps at times, resulting in the availability of substantial amounts of data to analyze. *Application design.* Augur is designed so that events are tracked and stored by a central logging contract with a high level of detail, which allowed insights in user behavior and simplified the extraction of data with ELF (in contrast to other DApps, in which logging is fragmented over multiple contracts). *Subsidiary information.* Information on Augur is widely available, such as in the Whitepaper [23], which served as basis, e.g., for conformance checking. Thus, Augur promised to be an interesting candidate for deeper analysis.

[2] https://twitter.com/AugurProject/status/1245715269042888706, accessed 2021-03-14.

[3] https://www.augur.net/blog/augur-v2-launch/, accessed 2021-03-14.

Markets are distinguished based on their outcome: *yes/no-markets* deal with binary questions, while *categorical* and *scalar markets* expect discrete and numeric answers, respectively. For each type, the Augur market process follows a procedure organized in four stages: market creation, trading, reporting, and settlement. *Market creation*: a market is set up for a future event, i.e., "market event." *Trading*: traders place bets on the outcome of the market by buying shares for that outcome. *Reporting*: a user reports the outcome of the event the market revolves around. The report can be challenged in *disputes* that are part of the reporting. *Settlement*: traders resolve their positions. Within these four stages, the Augur smart contracts specify 35 different types of events.

Participants can be active in an Augur market in five roles: market creator, traders, designated reporter, public reporters, and disputants. A *market creator* instantiates markets, including choosing a market question that revolves around a market event and appointing a designated reporter. *Traders* place bets by buying and selling shares of market outcomes. A *designated reporter* reports on the market event, thus creating the first tentative outcome of the market. If the designated reporter fails to submit a report within three days of the market event, reporting opens to *public reporters* who can report on the market outcome. Once an initial report is submitted, *disputants* can challenge the reported outcome of an event by crowd-sourcing a dispute bond with Augur's native token called "Reputation" (REP). If the dispute bond crosses a threshold, the crowd-funded outcome becomes the new tentative outcome. If disputes against an outcome remain unsuccessful, the tentative outcome becomes the final outcome. Depending on the dispute's success, disputants are redeemed after the dispute round (unsuccessful) or after the market is finalized (successful). A market finalizes if a tentative market outcome has not been successfully disputed within seven days. After it finalizes, market creators receive the market creation fee, designated reporters receive a fee if their report represents the final outcome, and traders settle their positions. As a final resolution mechanism for disputes, Augur also offers a fork event, which creates parallel instantiations of Augur based on each possible outcome of the forking market to which users can migrate. Forking is considered "very disruptive" and has not been triggered yet [23].

The Normative Process Model. Conformance checking requires a normative process model, which we created from information in the white paper [23]. We enriched it with information gained from discovery and conformance checking where the information in the white paper was not detailed or precise enough for our purposes. The resulting process model is shown in Fig. 2. Additional information on initial discrepancies is discussed in Sect. 5. We restricted the model to activities where the corresponding events were triggered.

3 Data Extraction and Pre-processing

On a second-generation blockchain like Ethereum, that allows for deploying and executing arbitrary smart contracts, log entries are the primary means for passing information to off-chain components. Commonly, log entries communicate

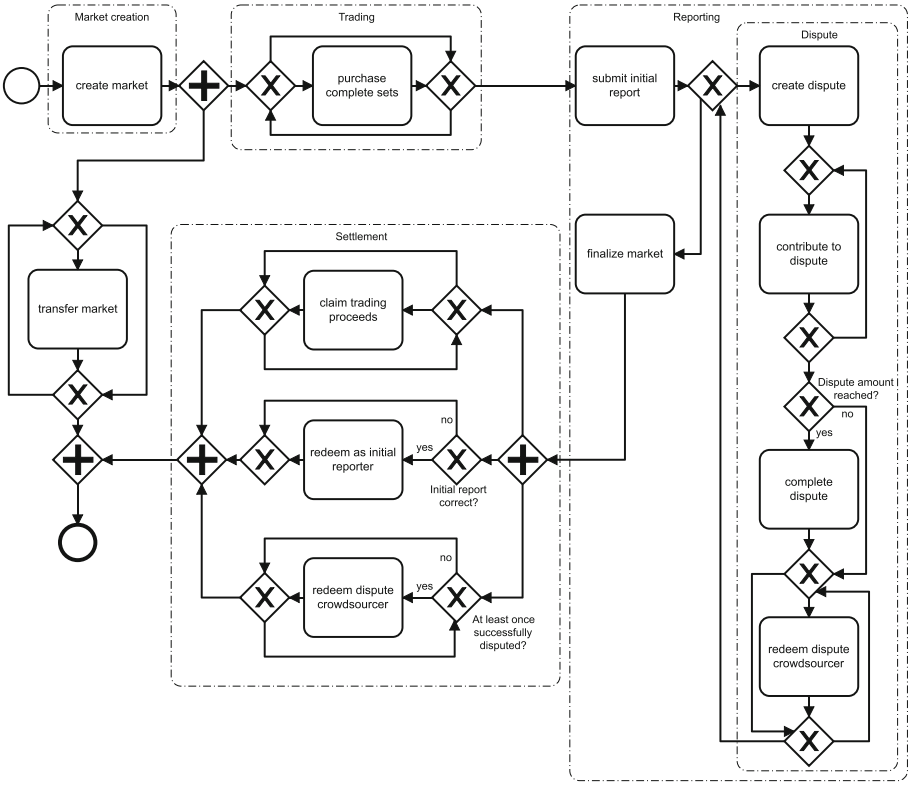

Fig. 2. Normative BPMN model.

information related to results of and events occurring during the execution of smart contract invocations. The developers of Augur v1.0 made extensive use of this feature and implemented a central logging contract that handles the emission of log entries. This contract defines a range of log event types intended to share detailed information about all possible events. Due to the level of detail provided by these log entries, we decided to solely focus on information from Augur's log entries for this case study. Hence, we typically did not consider additional information from the transactions or the states of Augur's smart contracts, as the information we could obtain this way is largely included in the log entries; deviations from this rule are marked.

We extracted the data using the publicly available Ethereum Logging Framework[4] [13,14]. ELF enables analysts to extract, transform, and format information from blocks, transactions, log entries, and smart contracts stored on Ethereum-based networks. ELF takes as input a manifest file, which contains instructions that define which data to extract and how to process it – see also Fig. 1. We defined such a manifest file for Augur v1.0 based on its source code,

[4] https://github.com/ChrisKlinkmueller/Ethereum-Logging-Framework.

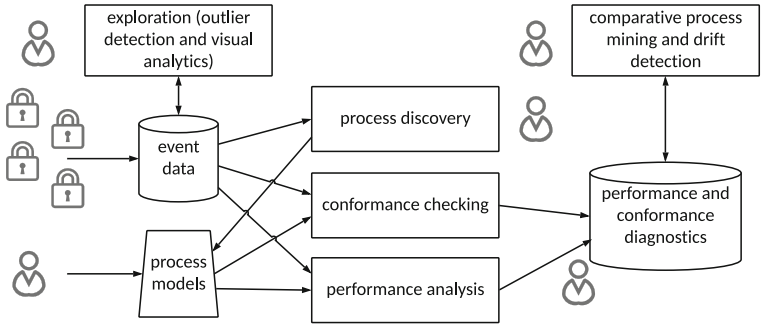

Fig. 3. Process mining techniques applied to the Augur v1.0 data extracted with ELF.

which provided us with the definitions for all the log entries. The execution of the manifest resulted in an event log in the XES format [4], where for each log entry, there is an XES event containing the information from Augur's log entry. We grouped the events into traces based on the notion of a market.

We extracted information related to 2897 markets stemming from the period of 2018-07-09 to 2020-11-10. The former date marks the date of the first execution of Augur v1.0. Regarding the latter, we ran the data extraction from 2020-11-12 to 2020-11-16, and the last event that we extracted was from 2020-11-10. However, as outlined in Sect. 2, the launch of Augur v2.0 in July 2020 rendered Augur v1.0 "economically insecure" and unsurprisingly caused a decline in user interest which already started after the announcement of Augur v2.0 in April 2020. To account for this decrease, we removed 162 cases that were either created after v2.0 was announced on 2020-04-02 or that were not finalized before its actual launch, leaving us with a total of 2735 cases and 22772 events.

For purposes of replication, all data and code used in this study are available publicly, including the source code of ELF (See Footnote 4), the manifest, the normative process model and the resulting XES log[5], the source code of Augur v1.0[6], the Augur white paper [23], and the data on the public Ethereum blockchain.

4 Process Mining Analysis and Results

As discussed in Sect. 3, we used ELF to extract an XES event log [4] for Augur v1.0. As a result, we can apply a range of process mining techniques, as illustrated by Fig. 3. It is possible to discover the actual betting/prediction process, check conformance of the process with respect to a normative model, analyze performance, and compare process variants [1]. In the remainder, we will mainly use the ProM process mining platform. We could apply any other process mining tool, e.g., open-source tools like PM4Py, Apromore, bupaR, and RapidProM, or

[5] https://github.com/ingo-weber/dapp-data.
[6] https://github.com/AugurProject/augur-core, accessed 2021-03-19.

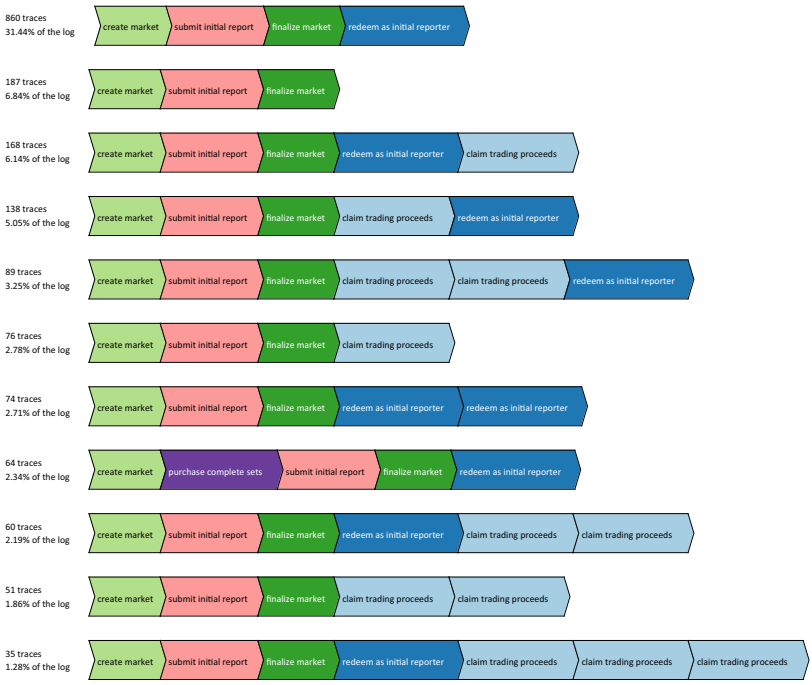

Fig. 4. Some of the most frequent variants, e.g., 860 markets follow the most frequent variant having only four events. The event data have a Pareto distribution with 35 variants (i.e., 8.5% of variants) explaining 80% of all cases.

commercial closed-source tools like Celonis, Disco, ProcessGold (UiPath), Minit, QPR, myInvenio, PAFnow, Lana, Software AG, Signavio (SAP), ABBYY Timeline, and Mehrwerk. However, our goal is not to present specific process mining algorithms or tools. Instead, we demonstrate that event data extracted from Ethereum using ELF can be used to analyze marketplaces like Augur.

4.1 Exploring the Event Data

All process mining tools start from event data [1]. An *event log* is a collection of events stored in a format like XES. An event may have many different attributes, but at least a *case identifier*, an *activity name*, and a *timestamp*. Additional attributes may refer to locations, resources, costs, transactional information, and on Ethereum blockchain, the consumed gas. Events are grouped using the case identifier and sorted using the timestamps. Hence, each case corresponds to a *trace*, i.e., a sequence of events. By focusing only on the activity names, these traces can be grouped into *variants*, i.e., sequences of activities. Most event logs have a Pareto distribution, i.e., a few variants explain a large proportion of the event log.

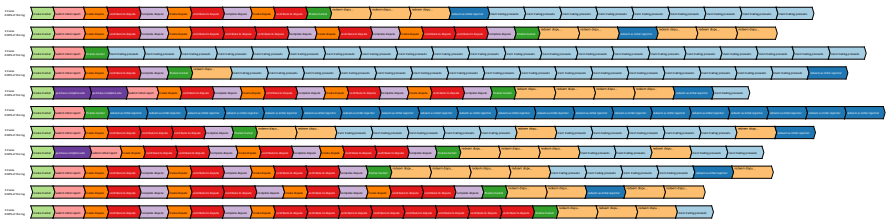

Fig. 5. Some of the unique variants that only have one corresponding market. The figure is not intended to be readable but gives an idea of the variability. 319 of the 414 variants are unique, covering 11.7% of all markets.

The event log we extracted and filtered as per Sect. 3 has 2735 cases (each case refers to a market), 22772 events, and 11 unique activities. There are 414 variants where 35 variants have at least ten corresponding cases and describe 2203 cases. This implies that 80% of the cases are described by less than 8.5% of variants. Some of the most frequent variants are shown in Fig. 4. 319 cases have a unique sequence of activities. A few of the shorter unique variants are shown in Fig. 5. The length varies from three to 226 events per case.

The event log contains 11 activities having the following frequencies: *claim trading proceeds* (6046), *redeem as initial reporter* (3259), *submit initial report* (2735), *create market* (2735), *finalize market* (2735), *contribute to dispute* (1598), *redeem dispute crowdsourcer* (1412), *create dispute* (901), *complete dispute* (780), *purchase complete sets* (570), and *transfer market* (1).

Figure 6 shows a so-called *dotted chart* where each dot refers to an event (i.e., 22772 dots). In a dotted chart, we can configure the two axes and the coloring of the dots [1]. In Fig. 6, the x-axis refers to the time of the event, the y-axis corresponds to the cases (i.e., markets) sorted by the time of the first event, and the color of the dot refers to the activity name (e.g., blue is the creation of the market). The dotted chart shows that many markets were created in the first

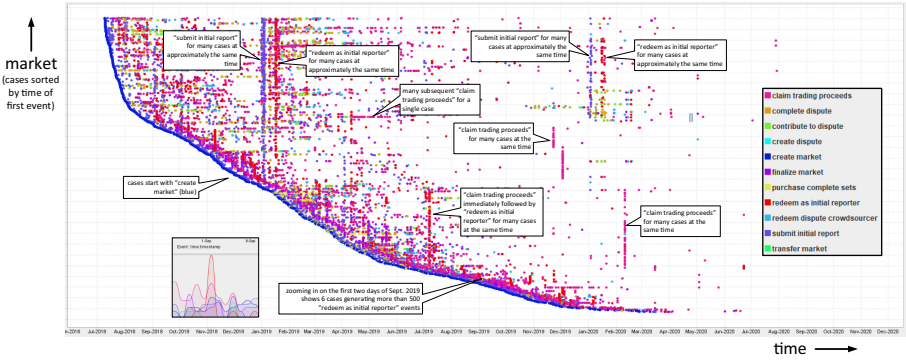

Fig. 6. Dotted chart showing all 22772 events. The vertical patterns indicate batching. (Color figure online)

month (July/August 2018). After that, there was a steady flow of new cases, until the arrival rate decreased after May 2019. The vertical patterns indicate *batching*, i.e., shorter periods where the same activity occurs for many cases. Some of these batching patterns are highlighted in Fig. 6. For example, on 2020-02-11, activity *claim trading proceeds* is executed 63 times for 53 cases in less than one hour. Another example of batching is the burst of the activities *claim trading proceeds* and *redeem as initial reporter* at the end of the day on 2019-07-07. These occur respectively 148 and 98 times in a three hours. There are also horizontal patterns indicating a sequence of events for the same case in a short period. For example, for the market "Ethereum Price at end of March 2019" we witnessed *redeem dispute crowdsourcer* and *claim trading proceeds* four and 132 times, respectively, within a period of a few weeks. Although one can already visually spot exceptional cases, we discuss these further when presenting the conformance checking results.

4.2 Process Discovery

Figures 7, 8, and 9 are based on the whole event log (i.e., 2735 markets generating 22772 events). Figure 7 shows a so-called Directly-Follows Graph (DFG) without filtering [1,2]. The nodes are activities and show the frequencies of each activity. The connections show how often one activity is followed by another. DFGs are the most-widely used discovery technique in commercial process mining tools due to their simplicity. However, there are several know problems, as demonstrated in [2]. These can be witnessed in Fig. 7, where there are many loops in the diagram because activities are not performed in a fixed order.

Figure 8 shows the Process Tree (PT) obtained by the Inductive Miner in ProM for the whole event log using the default settings [16]. The model is not intended to be readable, but one can see that the process model has more structure. 1771 of all cases (65%) can be explained by this model (the average trace fitness is 94%). Figure 9 shows the same model but now with timing information rather than frequencies. Two activities that have a longer sojourn time are highlighted.

Figures 10, 11, and 12 are based on the variants with at least ten corresponding cases. This filtered event log contains only 35 of the 414 variants; however, it represented over 80% of all markets (2203 cases). Due to the configurations used, all three models are guaranteed to be able to replay all 2203 cases from which these models were discovered. Actually, the process models in Figs. 11 and 12 can replay 2501 cases in the original event logs. Note that the PT was discovered using the basic Inductive Mining algorithm without further filtering [1,15]. This algorithm is also implemented in a few commercial systems (e.g., Celonis). The Petri net in Fig. 12 is semantically equivalent to the model in Fig. 11.

After focusing on the frequent variants, one can focus on particular parts of the process model. Such models are simpler and can be used to drill down. Let us, for example, focus on the dispute phase and consider only the activities *create dispute*, *contribute to dispute*, and *complete dispute*. Figures 13 and 14 show two process models explaining the dispute subprocess. This example illustrates that

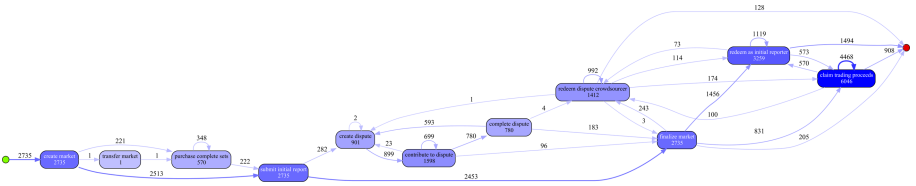

Fig. 7. Directly-Follows Graph (DFG) for the whole event log without filtering.

Fig. 8. Process Tree (PT) for the whole event log using default settings.

Fig. 9. Process Tree (PT) for the whole event log showing sojourn times.

process mining tools like ProM provide various ways to reduce the complexity and either focus on a particular part of the process or zoom-out (e.g., using aggregation) to see the overall process.

4.3 Conformance Checking and Unusual Cases

Figure 2 shows a normative process model that can be used for *conformance checking* [1,6]. The goal of conformance checking is to identify commonalities and differences between the modeled process and the actual process. Figures 15 and 16 show the reference model in the form of a process tree and a Petri net to allow for easy comparison with the discovered process models. A visual comparison shows that the reference model is close to the discovered models, but there are some notable differences. Compare, for example, Fig. 16 (Petri-net version of the reference model) with Fig. 12 (Petri-net able to replay all traces that happened at least ten times). Some of the striking differences: *transfer market* is missing in the discovered model (it was only executed in one trace); in the discovered process model, the activities *contribute to dispute* and *complete dispute* both occur precisely once after creating the dispute; and *redeem dispute crowdsourcer* can occur before *claim trading proceed* and *redeem as initial reporter*, but not after.

The representations shown in Figs. 15 and 16 can be used in ProM to perform a range of conformance checking techniques. Here, we limit ourselves to alignment-based conformance checking [1,6], i.e., for each trace in the event log, we searched for a path through the model that is closest.

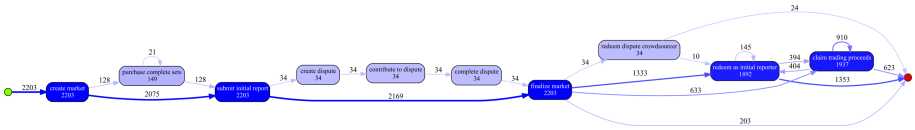

Fig. 10. Directly-Follows Graph (DFG) for 8.5% of variants covering 91.4% of the cases.

Fig. 11. Process Tree (PT) discovered for the filtered event log showing all paths.

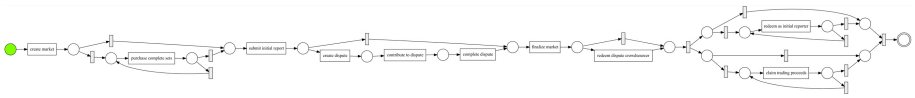

Fig. 12. The Petri net discovered for the filtered event log.

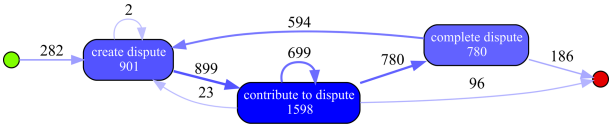

Fig. 13. Directly-Follows Graph (DFG) discovered for the dispute phase.

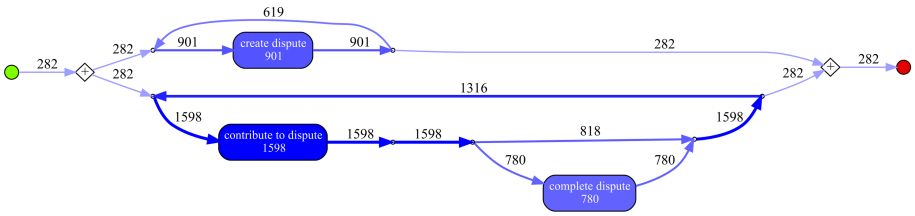

Fig. 14. Process Tree (PT) discovered for the dispute phase.

Fig. 15. Normative Process Tree based on the BPMN reference model.

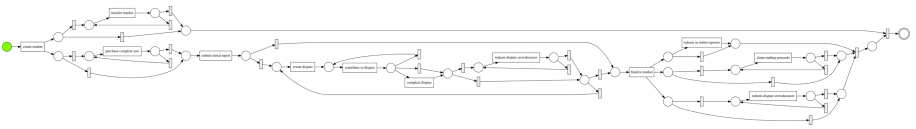

Fig. 16. Normative Petri net model.

Fig. 17. Replay results after aligning event logs and process model: 2511 of the 2735 cases are perfectly fitting the reference model.

ProM's diagnostics show that the reference model in Fig. 2 explains 2511 of the 2735 cases, i.e., 224 cases have at least one deviation. Figure 17 shows the same diagnostics after aligning event log and process model. There are 21647 synchronous moves (i.e., events in the log that fit the model) and 1125 moves on the log (i.e., events in the log that do not fit the model). However, there are no moves on the model (i.e., missing events). 223 of the 224 deviating cases have multiple *redeem as initial reporter* events, and 1119 of the 1125 log-only moves fall into this category. Almost all of these are instantaneous: using Disco's performance view, we see a median duration of 0 ms, a total duration over all 1119 moves of 42 days, and a maximum of 35 days – i.e., much of the whole duration can be accounted for with a single of the 1119 occurrences.

For a random sample of 20 of these 1119 occurrences, we inspected the underlying blockchain transactions, and observed the following pattern in all 20 instances: the first *redeem as initial reporter* event resulted in a payout, the second did not; the first and second transaction came from the same account in all 20 pairs; and the pairs were close together (between 0 and 47 blocks, the large majority with less than ten blocks). We also observed two cases with 108 *redeem as initial reporter* events ("Who will win the second democratic primary debate?" and "Will Tulsi Gabbard poll higher than Andrew Yang on August 12th?").

Like in discovery, there is the possibility to focus on selected parts of the process. Figure 18 shows conformance checking results for the dispute subprocess. There is only a single deviating case (see the upper part of Fig. 18) where there are two instances of two subsequent occurrences of create dispute without any

contributions in between. This is not possible according to the reference model. We discuss possible reasons for non-conformance in Sect. 5.

Next to the non-conforming cases, we identified the following *unusual cases*. In our data, we observed 13 cases with nine or more *complete dispute* events created in 2018. Ten of those were created in July 2018, the month the application was launched. The market with the highest number of contributions to disputes (98) was created on 2018-07-13 and posed the question, "Will the weather be good for the Bastille day military parade in Paris tomorrow?" The high ambiguity of the market question led to a debate in the Augur community revolving around wording of market questions and forking the application shortly after its launch. After 15 rounds of dispute[7], the market resolved as invalid and remained a familiar quotation. Soon after the debate around the Bastille Day market gained momentum, a meta-market was created ("Will the weather good when the "*Will the weather be good for the Bastille day military parade in Paris tomorrow?*" market resolve?"), betting on the events based on existing markets. That phenomenon, however, did not become a trend.

On 2018-07-10, four identical markets were created within 17 s, asking the question "Will Bitcoin go below $6000". One of the markets resolved as invalid after five weeks, while the other three went through dispute rounds until mid-September 2018, before also resolving invalid, all on the same day. In sum, this market question went through the highest number of dispute rounds (20).

4.4 Performance Analysis

Events have timestamps; therefore, it is trivial to enrich process models with timing information (e.g., waiting times and service times). This is a key capability of process mining and often used to improve operational processes, e.g., to reduce the time needed to produce a car or process a claim. For marketplaces like Augur, standard measurements like waiting times are less relevant, because the duration is related to the nature of the particular bet. For example, users can create markets for future events, no matter how far into the future the event is expected to take place. In this regard, we inspected the top 100 completed

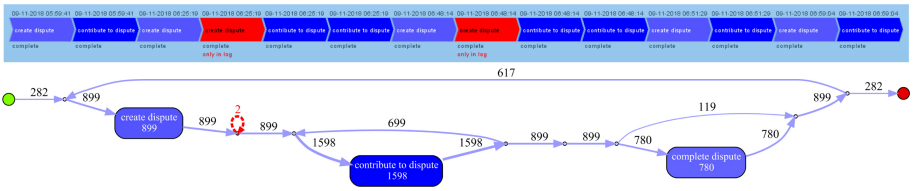

Fig. 18. Conformance checking results for the dispute phase including the activities *create dispute*, *contribute to dispute*, and *complete dispute*. Only one case is non-fitting.

[7] https://themajority.report/market/0x67ef420c045f3561d11ef94b24da7e2010650cc3, accessed 2021-03-05.

traces concerning the waiting time before submitting the initial report. All these markets were created in 2018 but referred to events that took place in 2019 and 2020. We also noted that there is a market with a prospected market event in 2070. In addition to these analyses, we zoomed into the dispute phase and only considered the activities *create dispute*, *contribute to dispute*, and *complete dispute* shown in Figs. 13 and 14. The mean duration of this phase is 15.4 days, the median duration is 7.1 days, and the longest duration is 111 days. This illustrates that time also plays a role in the analysis of markets.

5 Discussion

The Augur logging smart contract specifies 35 activities, of which only 11 activities could be observed after the data extraction. Partially, this discrepancy can be explained by events not being triggered throughout the application's life-cycle, such as the fork event. The option to fork serves as a final resolution mechanism in case a dispute could not be resolved over many rounds, and forms the last resort. Thus, not observing it was expected, and can indeed be seen as a sign that the incentive mechanism (geared towards avoiding forks) work. However, other events such as `DisputeWindowCreated` could be expected to be triggered frequently but were not part of the logged data, although specified in our ELF manifest.

Additionally, the white paper did not cover all events included in the smart contract (e.g., `DisputeCrowdsourcerCompleted` or `TradingProceedsClaimed`). That led to multiple iterations for creating the normative process model, where we started with the information in the white paper, ran discovery and conformance checking, found discrepancies, and resolved those by reconsidering the white paper and inspecting the source code. One observation was that the white paper in part turned out to be too abstract to model the normative process, as some information on the workings of Augur was not contained in it. For instance, `completeDispute` only happens if a sufficient amount of stakes is contributed to a dispute; this information is not contained in the white paper.

As pointed out in Sect. 4, cases with many dispute events were observed mainly in 2018, and mainly had creation dates in July 2018, the month of the Augur v1.0 launch. Disputes delay resolving a market and hint towards disagreement in the community. Their occurrence in the early days of the application indicates that the user group needed to build up experience in using the application. At times, users seemingly tested the resilience of the application (e.g., a market for "Did this market need a fork to be resolved?", created 2018-07-27, led to 12 complete dispute rounds but no fork). Eventually, the users learned to pose less ambiguous market questions, leaving less wiggle room for interpretation and reducing the potential for disputes.

Comparing the normative and discovered models in Fig. 16 and Fig. 12, we observed paths that were executed infrequently. Recall that the model in Fig. 12 was discovered from the 35 most frequent variants, and hence represents typical (but not all) observed behavior. Some paths, however, occurred very rarely if at

all: According to the white paper, after unsuccessful disputes crowdsourcers are redeemed at the end of the dispute round, while for successful disputes redeeming happens after the market finalized [23]. The sequence for redeeming unsuccessful disputes occurred only four times, and the log does not show a single case of *contribute to dispute* being directly followed by *redeem dispute crowdsourcer*. That poses the question why users did not make use of that option.

The most striking result from conformance checking was the frequent occurrences of *redeem as initial reporter* more than once, where we observed for a sample that the first event resulted in a payout and the second did not, and both originated from the same account within a short time frame. The logging on Augur could be made more precise here, and differentiate successful, legitimate transactions from others. Note that transaction inclusion may be subject to delay [30], and the timestamp for a transaction to be finally included in the ledger may be significantly after the transaction had been announced to the network. There are multiple possible explanations for the phenomenon of the repeated *redeem as initial reporter* events, including: (i) the reporter was impatient; (ii) the reporter used an automated tool with a time-out before retry, but the tool did not implement *retry* correctly (as per [30]); or (iii) the reporter tried to cheat or hack the system. Given that these attempts were unsuccessful and the reporter had to pay fees, and the same reporter accounts showed the same behavior repeatedly, we find (ii) the most plausible of the three scenarios.

The non-conforming repeated *create dispute* events happened in the same categorical market "2018 MLB World Series Champion". All four transactions (two pairs of two) were sent from the same blockchain account, and each pair was included in the ledger in direct succession in the same block. The two pairs were 95 blocks apart. The four transactions initialized four different dispute rounds, although at any time only one of those was active. By initializing future dispute rounds, the user "pre-staked" tokens for these future rounds. This was a *bug* in Augur v1.0, but turned out to be useful and was made a feature in v2.0, as we established in discussion with Augur's chief architect (see below).

Note that we did not aim to apply process mining as a design time or pre-deployment test for software vulnerabilities. However, we were able to show that process mining can serve as a tool to discover bugs and performance issues for blockchain applications post-deployment (based on actual user behavior), which enables developers to patch weaknesses or formalize unexpected behavior in updates. Methods for design time checks of vulnerabilities are nevertheless very important, particularly for DApps, but can be complemented with analyses such as ours.

To validate the veracity and assess the usefulness of the insights generated by our analyses, we interviewed Paul Gebheim, the chief architect of Augur. Given that we only interviewed one person, we classify results from this interview as *anecdotal evidence*; but given his position, we believe this evidence to be of value. We asked him to check assumptions we had – all of which he confirmed – and presented intermediate results from our analyses to him. From his perspective, using process mining for the analysis of blockchain applications generally, and Augur,

in particular, provides value in three ways. First, it helps to verify the design mechanisms and check for unintended behavior and bugs in the (immutable) code; immutability poses a challenge from a BPM perspective [20] and software engineering in general [28]. Second, process mining provides a clear view of how an application is used, which is also helpful for designing updated versions of an application. Third, it has great potential for technical and economic security analysis, e.g., in that, an auditor could create a model and conformance-check it against actual user behavior. Also, even though a smart contract typically implements a fixed set of rules, analyses of process variability may reveal valuable insights that could help evolve future versions of the smart contract, e.g., to align them better with changed user expectations.

The validity of this case study faced several threads to validity. As an internal thread, we might have introduced a bias in our conformance checking approach. As a basis for conformance checking, we used the entire normative process model (Fig. 2) and thus the overall control flow without checking the gate conditions for individual cases. That might have led to overly generalized results, ignoring non-conforming cases. Additionally, we largely observed the user behavior as a whole and not over time, which might have compromised awareness of maturation effects. An external thread to the study may be that the data we performed our analysis on was incomplete or its quality corrupted. We did, however, take precautions in reducing these threads by validating intermediate results and findings with Augur's user interface and their chief architect, as described above.

6 Conclusion and Future Work

In this paper, we conducted a case study on process mining for data extracted from the blockchain application Augur. To this end, we used ELF to extract data over essentially the entire lifecycle of Augur v1.0. We used process mining methods and tools to explore the data, discover models for a set of variants, and conducted conformance checking and performance analyses. Finally, we interviewed the chief architect of Augur to validate our insights and understand their usefulness. As stated in Sect. 3, we followed open science principles and made all data and code from our study available publicly.

In summary, we conclude that there is clear evidence for the usefulness of process mining on blockchain data. Main areas of interest for software developers may include user behavior analysis and security audits, for which we demonstrated the applicability of process mining tools. Indeed, we discovered a bug in Augur's smart contracts – albeit a non-critical one. Future research can be done evaluating other applications which might run on other blockchains, such as Hyperledger Fabric. The analysis method could be extended for blockchain-specific security and user studies, e.g., through drift detection and cohort analysis.

Acknowledgments. We are very thankful for the input of Paul Gebheim, chief architect at the Augur Project. We would also like to thank Martin Rebesky for writing the first version of the ELF manifest to extract an Augur event log.

References

1. van der Aalst, W.M.P.: Process Mining: Data Science in Action. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49851-4
2. van der Aalst, W.M.P.: A practitioner's guide to process mining: Limitations of the directly-follows graph. In: International Conference on Enterprise Information Systems (Centeris 2019), Procedia Computer Science, vol. 164, pp. 321–328 (2019)
3. van der Aalst, W.M.P., et al.: Business process mining: an industrial application. Inf. Syst. **32**(5), 713–732 (2007)
4. Acampora, G., Vitiello, A., Stefano, B., van der Aalst, W.M.P., Günther, C., Verbeek, E.: IEEE 1849: the XES standard. IEEE Comput. Intell. Mag. **12**(2), 4–8 (2017)
5. Andrews, R., Suriadi, S., Wynn, M., ter Hofstede, A.H.M., Rothwell, S.: Improving patient flows at St. Andrew's War Memorial Hospital's emergency department through process mining. In: Business Process Management Cases, pp. 311–333. Digital Innovation and Business Transformation in Practice (2018)
6. Carmona, J., Dongen, B., Solti, A., Weidlich, M.: Conformance Checking: Relating Processes and Models. Springer, Berlin (2018). https://doi.org/10.1007/978-3-319-99414-7
7. Corradini, F., Marcantoni, F., Morichetta, A., Polini, A., Re, B., Sampaolo, M.: Enabling auditing of smart contracts through process mining. In: ter Beek, M.H., Fantechi, A., Semini, L. (eds.) From Software Engineering to Formal Methods and Tools, and Back. LNCS, vol. 11865, pp. 467–480. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30985-5_27
8. De Weerdt, J., Schupp, A., Vanderloock, A., Baesens, B.: Process mining for the multi-faceted analysis of business processes - a case study in a financial services organization. Comput. Ind. **64**(1), 57–67 (2013)
9. Di Ciccio, C., Meroni, G., Plebani, P.: Business process monitoring on blockchains: potentials and challenges. In: Enterprise, Business-Process and Information Systems Modeling, pp. 36–51 (2020)
10. Di Ciccio, C., et al.: Blockchain-based traceability of inter-organisational business processes. In: Business Modeling and Software Design, pp. 56–68 (2018)
11. Duchmann, F., Koschmider, A.: Validation of smart contracts using process mining. In: Central European Workshop on Services and their Composition, pp. 13–16 (2019)
12. Jans, M., van der Werf, J.M., Lybaert, N., Vanhoof, K.: A business process mining application for internal transaction fraud mitigation. Expert Syst. Appl. **38**(10), 13351–13359 (2011)
13. Klinkmüller, C., Ponomarev, A., Tran, A.B., Weber, I., van der Aalst, W.M.P.: Mining blockchain processes: extracting process mining data from blockchain applications. In: BPM Blockchain Forum, pp. 71–86 (2019)
14. Klinkmüller, C., Weber, I., Ponomarev, A., Tran, A.B., van der Aalst, W.M.P.: Efficient logging for blockchain applications. CoRR abs/2001.10281 (2020). https://arxiv.org/abs/2001.10281, Accessed 21 Mar 2021
15. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Discovering block-structured process models from event logs - a constructive approach. In: Colom, J.-M., Desel, J. (eds.) PETRI NETS 2013. LNCS, vol. 7927, pp. 311–329. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38697-8_17
16. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Scalable process discovery and conformance checking. Softw. Syst. Model. **17**(2), 599–631 (2018)

17. Leemans, S.J.J., Poppe, E., Wynn, M.T.: Directly follows-based process mining: exploration a case study. In: 2019 International Conference on Process Mining, pp. 25–32 (2019)
18. Lemos, A.M., Sabino, C.C., Lima, R.M.F., Oliveira, C.A.L.: Using process mining in software development process management: a case study. In: 2011 IEEE International Conference on Systems, Man, and Cybernetics, pp. 1181–1186 (2011)
19. Mans, R., Schonenberg, M.H., Song, M., van der Aalst, W.M.P., Bakker, P.: Application of process mining in healthcare: a case study in a Dutch hospital. Biomed. Eng. Syst. Technol. **25**, 425–438 (2009)
20. Mendling, J., et al.: Blockchains for business process management - challenges and opportunities. ACM Trans. Manag. Inf. Syst. (TMIS) **9**(1), 41–416 (2018)
21. Mühlberger, R., Bachhofner, S., Di Ciccio, C., García-Bañuelos, L., López-Pintado, O.: Extracting event logs for process mining from data stored on the blockchain. In: Business Process Management Workshops, pp. 690–703 (2019)
22. Müller, M., Ruppel, P.: Process mining for decentralized applications (2019)
23. Peterson, J., Krug, J., Zoltu, M., Williams, A.K., Alexander, S.: Augur: A decentralized oracle and prediction market platform. Technical report, Forecast Foundation (2018). https://github.com/AugurProject/whitepaper/blob/master/v1/english/whitepaper.pdf, Accessed 05 Jan 2021
24. Reinkemeyer, L.: Process Mining in Action: Principles, Use Cases and Outlook. Springer, Berlin (2020). https://doi.org/10.1007/978-3-030-40172-6
25. Rovani, M., Maggi, F.M., Leoni, M., van der Aalst, W.M.P.: Declarative process mining in healthcare. Expert Syst. Appl. **42**(23), 9236–9251 (2015)
26. Rozinat, A., de Jong, I.S.M., Günther, C.W., van der Aalst, W.M.P.: Process mining applied to the test process of wafer scanners in ASML. IEEE Trans. Syst. Man Cybern. Part C **39**(4), 474–479 (2009)
27. Suriadi, S., Mans, R.S., Wynn, M.T., Partington, A., Karnon, J.: Measuring patient flow variations: a cross-organisational process mining approach. In: Asia Pacific Business Process Management, pp. 43–58 (2014)
28. Weber, I., Staples, M.: Programmable money: next-generation conditional payments using blockchain - keynote paper (2021)
29. Weber, I., Xu, X., Riveret, R., Governatori, G., Ponomarev, A., Mendling, J.: Untrusted business process monitoring and execution using blockchain. In: International Conference on Business Process Management, Rio de Janeiro, Brazil (2016)
30. Weber, I., et al.: On availability for blockchain-based systems (2017)
31. Xu, X., Weber, I., Staples, M.: Architecture for Blockchain Applications. Springer, Heidelberg (2019). https://doi.org/10.1007/978-3-030-03035-3