BlockConfess: Towards an Architecture for Blockchain Constraints and Forensics

Sabrina Kirrane
Institute for Information Systems & New Media
Vienna University of Economics and Business
Vienna, Austria
sabrina.kirrane@wu.ac.at

Claudio Di Ciccio
Department of Computer Science
Sapienza University of Rome
Rome, Italy
claudio.diciccio@uniroma1.it

***Abstract*—Although Blockchain is still an emerging technology it has the potential to serve as a general purpose information technology platform. Already, smart contract / chaincode platforms, such as Ethereum and Hyperledger Fabric, provide support for the execution of arbitrary computations. However, the suitability of these platforms for specifying and enforcing data and service usage constraints (e.g., usage policies, regulatory obligations, societal norms) and providing guarantees with respect to conformance has yet to be determined. In order to address this gap, in this position paper we argue that symbolic artificial intelligence techniques in the form of semantic technology based policy languages and business process conformance tools and techniques, can together be used to provide guarantees with respect to the behaviour of autonomous smart contract / chaincode applications.**

I. Introduction

Blockchain technology has the potential to disrupt service provision in both public and private sector organisations alike [28]. Already there are an array of different blockchain platform offerings: Bitcoin[1] is typically used for financial transactions; Permacoin[2] and Filecoin[3] enable untrusted parties to store each other's data; Ethereum[4] and Hyperledger Fabric[5] can support both the storage of data (i.e., state) and the execution of code (i.e., functions for expressing arbitrary computations), commonly referred to as smart contracts/chaincode (henceforth referred to simply as executable code). The latter, in particular, are especially suitable as a prospective general purpose Information and Communication Technology (ICT) architecture [39].

However, considering that this technology is still in its infancy, the suitability of blockchain executable code platforms for specifying and enforcing usage constraints (e.g., usage policies, regulatory obligations, and business rules) necessary to cater for the next generation of ICT applications, and for providing guarantees that both individual executable code applications and the blockchain system as a whole is acting in accordance with said constraints, has yet to be determined.

[1]https://bitcoin.org/en/
[2]https://github.com/input-output-hk/Scorex/wiki
[3]https://filecoin.io/
[4]https://www.ethereum.org/
[5] https://www.hyperledger.org/projects/fabric

In this context, there are two bodies of work which could potentially be adapted/extended to meet this need. When it comes from constraint specification and reasoning, the Semantic Web community have been actively working on using symbolic artificial intelligence techniques to allow for policy specification and enforcement (cf., [4, 20, 32]). Whereas from a compliance and conformance perspective the Business Process Management community have proposed techniques and tools for comparing and analysing traces generated by business process instances (cf., [6, 24]). Thus, in this position paper we discuss how said constraint specification and conformance checking techniques could together be used to support a-priori constraint enforcement and a-posteriori blockchain executable code verification.

Summarising our contributions, we: (i) propose a constraint and forensics framework that could be used to guide the development and enhancement of blockchain executable code platforms to support constraint specification, enforcement, and automatic conformance checking; and (ii) discuss how existing policy languages, business process management, and semantic encoding techniques could be combined and enhanced to enable blockchain technology to be used as a general purpose ICT platform.

The remainder of the paper is structured as follows: Section II presents the digital marketplace motivating scenario and the respective use case requirements used to guide our work. Section III provides the necessary background and related work on constraint specification, compliance checking, and techniques for ensuring software interoperability and reuse. Following on from this, Section IV sketches our blockchain constraints and forensics framework and Section V highlights open research questions in relation to semantic knowledge representation, compliance and conformance checking, and interoperability and reuse. Finally, we present our conclusions and plans for future work in Section VI.

II. Motivating Use Case Scenario

Although Blockchain executable code platforms have the capability to support use cases across a variety of domains (e.g., finance, supply chain, logistics, and healthcare) [39], the suitability of blockchain platforms for the specification of

Pre-print copy of the manuscript published by IEEE (available at ieeexplore.ieee.org)
identified by DOI: 10.1109/Blockchain50366.2020.00078

data and service constraints (e.g., usage policies, regulatory obligations, societal norms) needed to realise such use cases, remains an open research question. Additionally, considering the technology is still in its infancy, from a trust perspective there is a pressing need for blockchain executable code behaviour forensics, which is necessary to identify executable code non-conformity, faults, and malicious behaviour. Finally, from a general applicability perspective the proposed approach should be reuseable across a variety of different Blockchain executable code platforms.

The work is guided by requirements arising from a blockchain based digital marketplace, such as that envisaged in [37]. Here blockchain executable code could be used not only to publish and consume data, but also to perform transactions in digital data markets without human intervention. For such digital data markets to be plausible, it is crucial that data publishers can specify who gets access to the data under what conditions, and data consumers can specify their requirements in a manner that allows the marketplace to find the most relevant detests. Given the variety of policies needed (e.g., usage conditions, requests requirements, privacy policies, regulatory conditions, offers, agreements), the marketplace use case it particularly suitable for guiding our analysis. Here, constraints could be used to mediate between buyers and sellers (based on pre-specified requirements), for instance allowing for the exchange and storage of machine readable requests, offers, agreements, possibly involving negotiation of terms and conditions. Whereas, forensic techniques are needed to ensure that the market is working as expected.

Our hypothesis is that blockchain executable code constraints, in the form of usage policies, regulatory obligations, business rules, and societal norms, together with executable code behavioural forensics, will allow companies to develop innovative services on top of blockchain executable code platforms. Starting from this hypothesis, we devise three primary research questions:

i) Which semantic knowledge representation techniques are needed to describe data and service constraints, as well as relevant metadata, in a manner that supports automatic enforcement via the executable code?
ii) How can existing business process conformance tools and techniques be modified/adjusted to automatically detect unexpected executable code behaviour (i.e., non conformity, faults, malicious behaviour) in a distributed setting?
iii) How do we ensure that both the proposed constraints and forensic tools and techniques are designed in a manner that supports interoperability and reusability across various blockchain platforms?

The above research questions translate into four concrete objectives:

i) Analyse the expressivity of existing blockchain executable code in terms of constraint specification and enforcement, by performing preliminary forensics over existing smart contracts / chaincode (i.e. preliminary data), using an attacker approach;
ii) Build upon existing policy languages with formal semantics, propose representations that are suitable for specifying executable code data and service usage constraints, and a framework for policy-based reasoning over a variety of constraints, paying particular attention to balancing expressivity and scalability;
iii) Adapt/extend existing platform analysis and process conformance tools and techniques, such that they can be used for automatic executable code monitoring and forensics; and
iv) Develop a blockchain data and query execution marketplace prototype in order to demonstrate the effectiveness of the proposed constraint-aware executable code monitoring and forensics from a robustness perspective (i.e., correctness, safety, performance and scalability).

III. Background & Related Work

In this section, we present the necessary background and related work on constraint specification, behavioural forensics, interoperability, and reusability, that together are needed in order to support constraint enforcement and automated compliance and conformance checking.

A. Constraint specification and enforcement

From a constraint specification and enforcement perspective, there has been work on extending blockchain technology to keep track of both data and access transactions [40], and other work which demonstrates how blockchain can be used to record access policies and rights transfer in an auditable manner [25]. Both Idelberger et al. [17] and Governatori et al. [15] argue that declarative logic based languages are more suitable for representing legal contracts as code than the procedural languages that are commonly used to develop blockchain applications. More specifically, Idelberger et al. [17] demonstrate the effectiveness of logic based contracts from a constraint lifecycle perspective, while Governatori et al. [15] assess the legal and technical challenges associated with using blockchain smart contracts as legally binding agreements.

Beyond the blockchain domain, there is a mature body of work on general policy languages, with formal semantics that can be used to establish the correctness and safety of the encoded policies. For instance, Rei [20] and Protune [4] use ontologies to represent concepts, the relationships between these concepts and the evidence needed to prove their truth, and rules to represent policies. When it comes to standardisation initiatives, the Open Digital Rights Language (ODRL) [32] was published as a recommendation by the World Wide Web Consortium (W3C) in 2018. Although ODRL was primarily intended to define rights to or to limit access to digital resources (cf. [33]), it has demonstrated its potential as a general policy language, for instance for expressing: access policies [36]; requests, data offers and agreements [35]; and basic regulatory policies [10].

Pre-print copy of the manuscript published by IEEE (available at ieeexplore.ieee.org)
identified by DOI: 10.1109/Blockchain50366.2020.00078

B. Compliance and conformance checking

From a behavioural forensics perspective, Magazzeni et al. [26] identify a set of questions that can be used to guide the development of automated validation and verification techniques. In particular, they focus on how to guarantee a correspondence between the actual runs of executable code and the intended, to-be behavior. A number of approaches have been proposed since that also proceed in that direction. When it comes to logic-based approaches, Chen et al. [8] propose a language-independent approach to smart contract verification, however both an investigation into the general applicability and the development of automated verification tools based on the proposed approach are left to future work. Wang et al. [38] put forward an approach that can be used to check whether a contract correctly implements the underlying workflow policy expressed as a finite state machine. Harz and Knottenbelt [16] provide a survey of existing tools and techniques used for smart contract verification. The authors conclude the article by identifying the need for further work encoding the formal semantics of smart contract such that it is possible to develop automated verification tools.

In addition to works that focus specifically on smart contract verification, existing work focusing on platform analytics and process conformance checking provide interesting starting points for advancing this field of research. For instance, the analytical approach proposed by Ali-Gombe et al. [1] could be adapted to cater for constraint-based process discovery over blockchain transactions, to identify patterns of malicious behaviours, faults, and execution nonconformity of processes carried out on-chain. In addition, there is a mature body of work on process conformance checking that provides techniques and methods for comparing and analysing observed instances [6]. Of particular relevance in that sense is the continuous auditing framework, which applies data mining and process mining together with human expertise, to the task of transaction verification [18]. As a basis for those techniques to be applied, Di Ciccio et al. [11] illustrate mechanisms that can be used to track process executions via transactions recorded on the blockchain.

C. Interoperabiliy and reusability

From an interoperabiliy and reusability perspective, Lamparelli et al. [22] highlight the need for interoperability from both a platform and an application perspective. The authors propose a smart contract description language that can be used to describe smart contract interfaces for both permissioned and permissionless blockchains. While, Falazi et al. [14] propose a mechanism that can be used to combine smart contracts in the form of a process that is needed in order to complete a task. Liu et al. [23] in turn propose several design patterns, which they classify as creational, structural, inter-behavioral and intra-behavioral.

More broadly, syntactic and semantic annotation and completion techniques are commonly used to help developers to implement reusable code, however they are typically tied to the development context and specific to a certain language. For instance, the Java Modeling Language (JML) can be used to describe Java functions [7], while others propose mechanisms to automatically create function definitions based on metadata supplied by the developer [31]. In terms of semantic modelling the PROV ontology [29] is a well known vocabulary used to represent provenance information. Other work employing a semantic based approach focuses specifically on semantically declaring and describing functions and the metadata of their implementations [9].

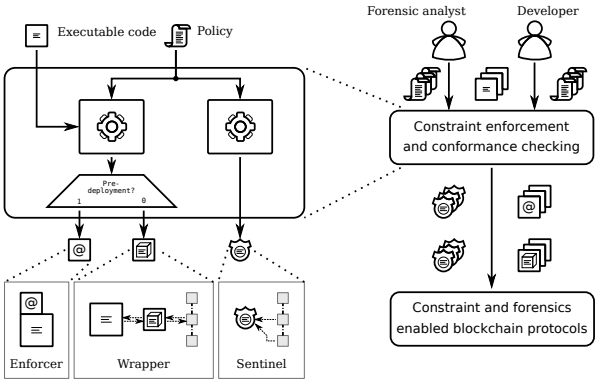

Figure 1: BlockConfess constraint and forensics framework.

In this paper, we discuss how such approaches could be combined in order to develop and enhance the technical foundations of blockchain executable code platforms to support constraint specification, enforcement, and automatic conformance checking.

IV. Approach

Figure 1 sketches our conceptual constraint and forensics framework. Both policies and executable code are capable of expressing *constraints*. The former are declarative specifications of behaviour, whereas the latter is an implementation mode. We assume two stages for constraint verification, depending on the run-time phase: (ex-ante) *enforcement* and (ex-post) *conformance checking*. In addition, we identify two categories of system-administrators (that are not specific to any particular use case): *developers*, who are responsible for developing executable code and wish to verify that the executable code respects given policies; and *forensic analysts*, who are interested in identifying behaviours that could denote security breaches, malicious activities, or potential vulnerabilities.

A. Constraint enforcement and conformance checking

The constraint enforcement and conformance checking component accepts as input constraints in the form of *executable code* and *policies*. Executable code applications encode business rules, namely the behavioural constraints that the code must respect, and policies denote desired properties and characteristics offered by the blockchain application

Pre-print copy of the manuscript published by IEEE (available at ieeexplore.ieee.org)
identified by DOI: 10.1109/Blockchain50366.2020.00078

under development or already operating. From a forensics perspective we are interested in scenarios where the focus is not on the verification of single instances of executable code, but rather ensembles thereof.

Enforcers are needed to tackle the problem of a-priori constraint enforcement. If the code has not yet been deployed, the policies need to be turned into encoded statements asserting properties of the code at hand, in a manner that restricts its behaviour thereby enabling automated compliance. We thus name those encoded statements *enforcers*.

Wrappers also deal with a-priori constraint enforcement, although they are used when executable code is already deployed. *Wrappers* act as a façade, intercepting and re-routing the messages and transactions, such that deployed applications can also benefit from the envisaged conformance checking.

Sentinels focus on a-posteriori verification of blockchain executable code based on their recorded transactions. Therefore, they query (ex-post) or capture (at runtime) the transaction flow into and out of the executable code under analysis. Hence the name, *sentinels*.

B. Constraint and forensics enabled blockchain protocols

The constraint and forensics enabled blockchain protocols component denotes changes required to the fabric of existing blockchain infrastructures, such that it is possible to use output produced by enforcers, wrappers, and sentinels to provide a point-in-time overview of individual executable code applications, a subset of executable code applications within the network, or the network as a whole. Here, semantic knowledge representation techniques are needed in order to encode constraints and events in an interoperable manner, and forensic techniques are needed in order to collect and synthesise constraints and events, and to develop conformance checking algorithms that will work in a distributed setting. In particular, we envisage a forensics protocol used by peers that are responsible for monitoring and conformance checking, similar to the existing proof-of-work protocol, although at the higher layer of abstraction of executable code.

V. Discussion

When it comes to the realisation of the proposed blockchain based constraints and forensics framework there are several open research questions with respect to: the suitability of existing policy languages for specifying executable code constraints in a machine readable manner; the effectiveness of existing business process management conformance checking techniques in terms of this still emerging technology; and ensuring the interoperability and reusability of the various tools and technologies.

A. Constraint specification and enforcement

From a constraints perspective, there is a need for a policy language that can be adapted/extended such that it is possible to express a variety of policies (e.g., usage conditions, requests requirements, privacy policies, regulatory conditions, offers, agreements). Here, general policy languages such as Rei [20] and Protune [4], coming from the semantic Web research community, are particularly interesting due to both generality and formal underpinnings. Rei polices are based on deontic logic, thus it is possible to model not only permissions and prohibitions, but also obligations and dispensations. The Rei policy language leverages the W3C Web Ontology Language [27] for policy specification, and N3 rules for policy enforcement [19]. Protune policies are specified using rules and meta rules that constrain the behaviour of web agents and the usage of web resources. One of the primary goals of the language is to support the exchange of information between negotiating peers [3]. However, neither Rei nor Protune managed to achieve widespread adoption. In contrast, the W3C Open Digital Rights Language (ODRL) [32] recommendation has been gaining traction in recent years. Primary building blocks include the ODRL Information Model[6] and the ODRL Vocabulary & Expression[7] specifications. The profile based extension mechanism is particularly interesting as it allows for the language to be adapted for different use cases (e.g., access control, usage control, legal policies). One of the major limitations of the language is the fact that there is no underpinning formal semantics.

When it comes to our automated digital marketplace scenario, it would be good to leverage the OWL based policies provided for by Rei, the negotiation capabilities of Protune, and the flexible profile based extension mechanism proposed by the ODRL community group. However, their combination is not a trivial task due to different or missing underlying formalisms. Additionally, their suitability for expressing the policies needed for an autonomous data marketplace and their adaption to a blockchain setting are open research questions. Also, it is unclear if these policy languages can deal the volume, velocity, variety, and veracity of data we are faced with today, thus we need to be mindful of the need to balance expressiveness and computational complexity.

B. Compliance and conformance checking

From a platform analysis and business process conformance checking perspective, it may be possible to adapt existing business process management tools and techniques [6] so that they can be used for automatic executable code monitoring and forensics in a distributed setting. Conformance checking compares the behaviour of a normative model to the one evidenced by execution traces. Constraints and policies can be interpreted as a normative model to be checked against blockchain data. However, the extraction, transformation, and loading of data from the blockchain is a non-trivial task [11]. Preliminary results to that end are presented by the recent approaches of Mühlberger et al. [30]

[6]https://www.w3.org/TR/odrl-model/
[7]https://w3c.github.io/poe/vocab/

Pre-print copy of the manuscript published by IEEE (available at ieeexplore.ieee.org)
identified by DOI: 10.1109/Blockchain50366.2020.00078

and Klinkmüller et al. [21]. However, we see the checking of constraints on mixed on-chain and off-chain data as an avenue for future investigations. We envision that novel solutions could benefit from the advancements on ontology-based data access approaches such as that of Calvanese et al. [5] and from novel object-centric specification languages for processes (e.g., OCBC [2]) to represent the stateful nature of transactional objects handled by executable code. On top of that, semantic technologies could be leveraged to query and reason on the retrieved information for conformance checking [12]. The cross-validation of information retrieved from the blockchain and the real-world is another open investigation [13]. Furthermore, given that compliance and conformance checking tools are usually designed for in-house business process execution, it remains to be seen if they are effective and what adaptations are needed when it comes to executable code conformance checking.

C. Interoperabilily and reusability

By employing semantic technologies to encode constraints specified in the form of polices, annotations associated with executable code, and provenance events stored in the blockchain ledger, it will be possible to support not only interoperabilily and reusability across blockchain platforms, but also traceabiliy between events in a manner that facilitates automatic compliance checking. In addition to the ODRL vocabulary mention earlier, there are a number of existing vocabularies that can be adapted/extended to satisfy our needs. When it comes to encoding provenance information related to constraints, processes, logs, etc... the W3C PROV Ontology[8] is an obvious choice. As for temporal expressions the W3C Spatial Data on the Web Working Group has recently proposed the OWL-Time Ontology[9] as a candidate recommendation. Additionally, there are a number of approaches for representing and reasoning over events, such as the Event Ontology[10] and the Content Ontology Design Pattern [34], which would be particularly useful when it comes to encoding and reasoning over processes. In order to make executable code interface declarations more accessible, the approach adopted by De Meester et al. [9] could be used to describe both the function and the metadata of their implementations.

When it comes to interoperabilily and reusability one of the primary challenges will be implementing the changes required into the fabric of existing blockchain infrastructures, such that it is possible for executable code applications to share and reason over service offerings, usage policies, and event logs. Another level of complexity is added by the fact that the proposed tools and techniques need to work in a variety of different Blockchain executable code platforms.

[8]https://www.w3.org/TR/prov-overview/

[9]https://www.w3.org/TR/owl-time/

[10]http://motools.sourceforge.net/event/event.html

VI. Conclusion

In this position paper, we argue that executable code constraints, in the form of usage policies, regulatory obligations, business rules, and societal norms, together with executable code behavioural forensics are needed in order to unleash the potential of blockchain technology as a general purpose ICT architecture. We proposed a blockchain based constraints and forensics framework, comprising three different constraint enforcement and conformance checking subcomponents, and identified the need for changes to the fabric of existing blockchain infrastructures in order to support automated enforcement, compliance and conformance checking. When it comes to the instantiation of the proposed framework we discussed how semantic knowledge representation together with business process management compliance and conformance checking techniques could be used to specify constraints, interface declarations, and event logs, such that it is possible to support automatic compliance and conformance checking. Future work includes the conceptualisation of the ideas presented herein in the form of a blockchain data marketplace, and the corresponding proof of concept, which will be used to validate the effectiveness of the proposed constraint and forensics enabled blockchain protocols from a robustness perspective (i.e., correctness, safety, performance and scalability).

Acknowledgements: This work was partly supported by the Austrian Science Fund (FWF) and netIdee SCIENCE under grant V 759, the COMET centre ABC – Austrian Blockchain Center, and the Italian MIUR under grant "Dipartimenti di eccellenza 2018-2022" of the Department of Computer Science at Sapienza University of Rome. We would like to thank Ruben Verborgh and Anastasia Dimou for their support in shaping some of the ideas presented in this paper.

References

[1] A. I. Ali-Gombe, B. Saltaformaggio, D. Xu, G. G. Richard III, et al. Toward a more dependable hybrid analysis of android malware using aspect-oriented programming. *Computers & security*, 73, 2018.

[2] A. Artale, D. Calvanese, M. Montali, and W. M. P. van der Aalst. Enriching data models with behavioral constraints. In *Ontology Makes Sense*, volume 316 of *Frontiers in Artificial Intelligence and Applications*, 2019.

[3] P. Bonatti, J. De Coi, D. Olmedilla, and L. Sauro. Protune: A rule-based provisional trust negotiation framework. 2010.

[4] P. A. Bonatti and D. Olmedilla. Rule-based policy representation and reasoning for the semantic web. In *Reasoning Web Summer School*, 2007.

[5] D. Calvanese, T. E. Kalayci, M. Montali, and S. Tinella. Ontology-based data access for extracting event logs from legacy data: The onprom tool and methodology. In *BIS*, 2017.

Pre-print copy of the manuscript published by IEEE (available at ieeexplore.ieee.org)
identified by DOI: 10.1109/Blockchain50366.2020.00078

[6] J. Carmona, B. van Dongen, A. Solti, and M. Weidlich. *Conformance Checking: Relating Processes and Models*. Springer, 2018.

[7] P. Chalin, J. R. Kiniry, G. T. Leavens, and E. Poll. Beyond assertions: Advanced specification and verification with JML and esc/java2. 4111, 2005.

[8] X. Chen, D. Park, and G. Roşu. A language-independent approach to smart contract verification. In *ISoLA (4)*, 2018.

[9] B. De Meester, T. Seymoens, A. Dimou, and R. Verborgh. Implementation-independent function reuse. *Future Generation Computer Systems*, 2019.

[10] M. De Vos, S. Kirrane, J. Padget, and K. Satoh. ODRL policy modelling and compliance checking. In *In Rules and Reasoning*, 2019.

[11] C. Di Ciccio, A. Cecconi, J. Mendling, D. Felix, D. Haas, D. Lilek, F. Riel, A. Rumpl, and P. Uhlig. Blockchain-based traceability of inter-organisational business processes. In *BMSD*, 2018.

[12] C. Di Ciccio, F. J. Ekaputra, A. Cecconi, A. Ekelhart, and E. Kiesling. Finding non-compliances with declarative process constraints through semantic technologies. In *CAiSE Forum*, 2019.

[13] C. Di Ciccio, G. Meroni, and P. Plebani. Business process monitoring on blockchains: Potentials and challenges. In *BPMDS/EMMSAD@CAiSE*, 2020.

[14] G. Falazi, M. Hahn, U. Breitenbücher, F. Leymann, and V. Yussupov. Process-based composition of permissioned and permissionless blockchain smart contracts. In *EDOC*, 2019.

[15] G. Governatori, F. Idelberger, Z. Milosevic, R. Riveret, G. Sartor, and X. Xu. On legal contracts, imperative and declarative smart contracts, and blockchain systems. *Artificial Intelligence and Law*, 26(4), 2018.

[16] D. Harz and W. Knottenbelt. Towards safer smart contracts: A survey of languages and verification methods. *arXiv preprint arXiv:1809.09805*, 2018.

[17] F. Idelberger, G. Governatori, R. Riveret, and G. Sartor. Evaluation of logic-based smart contracts for blockchain systems. In *RuleML*, 2016.

[18] M. Jans and M. Hosseinpour. How active learning and process mining can act as continuous auditing catalyst. *Int. J. Account. Inf. Syst.*, 32, 2019.

[19] L. Kagal and T. Berners-lee. Rein : Where policies meet rules in the semantic web. Technical report, Laboratory, Massachusetts Institute of Technology, 2005.

[20] L. Kagal, T. Finin, and A. Joshi. A policy language for a pervasive computing environment. In *Proceedings POLICY 2003. IEEE 4th International Workshop on Policies for Distributed Systems and Networks*, 2003.

[21] C. Klinkmüller, A. Ponomarev, A. B. Tran, I. Weber, and W. van der Aalst. Mining blockchain processes: Extracting process mining data from blockchain applications. In *BPM (Blockchain and CEE Forum)*, 2019.

[22] A. Lamparelli, G. Falazi, U. Breitenbücher, F. Daniel, and F. Leymann. Smart contract locator (scl) and smart contract description language (scdl). In *ICSOC*, 2019.

[23] Y. Liu, Q. Lu, X. Xu, L. Zhu, and H. Yao. Applying design patterns in smart contracts. In *ICBC*, 2018.

[24] L. T. Ly, F. M. Maggi, M. Montali, S. Rinderle-Ma, and W. M. P. van der Aalst. Compliance monitoring in business processes: Functionalities, application, and tool-support. *Inf. Syst.*, 54, 2015.

[25] D. D. F. Maesa, P. Mori, and L. Ricci. A blockchain based approach for the definition of auditable access control systems. volume 84, 2019.

[26] D. Magazzeni, P. McBurney, and W. Nash. Validation and verification of smart contracts: A research agenda. *Computer*, 50(9), 2017.

[27] D. L. McGuinness and F. van Harmelen. OWL Web Ontology Language Overview. Technical report, W3C, February 2004.

[28] J. Mendling, I. Weber, W. V. D. Aalst, J. V. Brocke, et al. Blockchains for business process management - challenges and opportunities. *ACM Trans. Manage. Inf. Syst.*, 9(1), 2018.

[29] P. Missier, K. Belhajjame, and J. Cheney. The W3C PROV family of specifications for modelling provenance metadata. In *EDBT*, 2013.

[30] R. Mühlberger, S. Bachhofner, C. Di Ciccio, L. García-Bañuelos, and O. López-Pintado. Extracting event logs for process mining from data stored on the blockchain. In *BPM Workshops*, 2019.

[31] K. I. Murray and J. P. Bigham. Beyond autocomplete: Automatic function definition. In *VL/HCC*, 2011.

[32] ODRL Community Group. ODRL Information Model 2.2. Technical report, W3C, June 2018.

[33] O. Panasiuk, S. Steyskal, G. Havur, A. Fensel, and S. Kirrane. Modeling and reasoning over data licenses. In *ESWC (Satellite Events)*, 2018.

[34] M. Rinne, E. Blomqvist, R. Keskisärkkä, and E. Nuutila. Event processing in RDF. In *WOP*, 2013.

[35] S. Steyskal and S. Kirrane. If you can't enforce it, contract it: Enforceability in policy-driven (linked) data markets. In *Semantics (posters & demos)*, 2015.

[36] S. Steyskal and A. Polleres. Defining expressive access policies for linked data using the odrl ontology 2.0. In *SEMANTICS*, 2014.

[37] H. Subramanian. Decentralized blockchain-based electronic marketplaces. *Commun. ACM*, 61(1), 2018.

[38] Y. Wang, S. K. Lahiri, S. Chen, R. Pan, I. Dillig, C. Born, I. Naseer, and K. Ferles. Formal verification of workflow policies for smart contracts in azure blockchain. In *VSTTE*, 2019.

[39] X. Xu, I. Weber, and M. Staples. *Architecture for Blockchain Applications*. Springer, 2019. ISBN 978-3-030-03034-6.

[40] G. Zyskind, O. Nathan, et al. Decentralizing privacy: Using blockchain to protect personal data. In *IEEE Security and Privacy Workshops*, 2015.

Pre-print copy of the manuscript published by IEEE (available at ieeexplore.ieee.org)
identified by DOI: 10.1109/Blockchain50366.2020.00078

This document is a pre-print copy of the manuscript
(Kirrane and Di Ciccio 2020)
published by IEEE (available at ieeexplore.ieee.org).

The final version of the paper is identified by DOI: 10.1109/Blockchain50366.2020.00078

References

Kirrane, Sabrina and Claudio Di Ciccio (2020). "BlockConfess: Towards an Architecture for Blockchain Constraints and Forensics". In: *AIChain@Blockchain*. IEEE, pp. 539–544. ISBN: 978-0-7381-0495-9. DOI: 10.1109/Blockchain50366.2020.00078.

BibTeX

```
@InProceedings{   Kirrane.DiCiccio/AIChain2020:BlockConfess,
  author        = {Kirrane, Sabrina and Di Ciccio, Claudio},
  title         = {{B}lock{C}onfess: {T}owards an Architecture for Blockchain
                   Constraints and Forensics},
  booktitle     = {AIChain@Blockchain},
  year          = {2020},
  pages         = {539-544},
  crossref      = {AIChain2020},
  doi           = {10.1109/Blockchain50366.2020.00078},
  keywords      = {Blockchain; Conformance Checking; Policies; Semantic
                   technologies; Process rules}
}
@Proceedings{     AIChain2020,
  title         = {{IEEE} 2nd International Conference on Blockchain,
                   Blockchain 2020, Rhodes Island, Greece, November 02-06,
                   2020},
  year          = {2020},
  publisher     = {IEEE},
  isbn          = {978-0-7381-0495-9}
}
```