

# Blockchain Technology for Collaborative Information Systems

Edited by

Marlon Dumas<sup>1</sup>, Richard Hull<sup>2</sup>, Jan Mendling<sup>3</sup>, and Ingo Weber<sup>4</sup>

1 University of Tartu, EE, marlon.dumas@ut.ee

2 IBM TJ Watson Research Center – Yorktown Heights, US, hull@us.ibm.com

3 Wirtschaftsuniversität Wien, AT, jan.mendling@wu.ac.at

4 Data61, CSIRO – Sydney, AU, ingo.weber@data61.csiro.au

---

## Abstract

Blockchain technology enables an evolving set of parties to maintain a safe, permanent, and tamper-proof ledger of transactions without a central authority. This technology opens manifold opportunities to redesign business-to-business collaborations, while bringing about numerous challenges. These opportunities and challenges were discussed in the Dagstuhl Seminar 18332 “Blockchain Technology for Collaborative Information Systems”. This report documents the program and the outcomes of the seminar.

**Seminar** August 12–17, 2018 – <http://www.dagstuhl.de/18332>

**2012 ACM Subject Classification** Applied computing → Business process management, Information systems → Collaborative and social computing systems and tools

**Keywords and phrases** Blockchain, BPM, Business Collaboration, Commerce, Logistics, Business Models (economic), Smart Contracts, Privacy

**Digital Object Identifier** 10.4230/DagRep.8.8.67

## 1 Executive summary

*Marlon Dumas (University of Tartu, EE)*

*Richard Hull (IBM TJ Watson Research Center – Yorktown Heights, US)*

*Jan Mendling (Wirtschaftsuniversität Wien, AT)*

*Ingo Weber (Data61, CSIRO – Sydney, AU)*

**License** © Creative Commons BY 3.0 Unported license  
© Marlon Dumas, Richard Hull, Jan Mendling, and Ingo Weber

Blockchain technology enables an evolving set of parties to maintain a safe, permanent, and tamper-proof ledger of transactions without a central authority. This technology opens manifold opportunities to redesign Business-to-Business (B2B) collaborations in a wide range of fields, including supply chain, logistics, service agreements, healthcare, and Industry 4.0. Importantly, it can enable substantial efficiency gains in terms of cost and time it takes to set-up and perform collaborative processes, particularly in settings where there is a lack of trust between the parties involved in the collaboration. Traditionally, collaborative processes are executed by relying on trusted third-party providers such as Electronic Data Interchange (EDI) hubs or escrows. This centralized architecture creates entry barriers and hinders bottom-up innovation. Blockchains and smart contracts enable these processes to be executed in a distributed manner without delegating trust to central authorities nor requiring mutual trust between each pair of parties. Further, blockchain enables fine-grained access



Except where otherwise noted, content of this report is licensed under a Creative Commons BY 3.0 Unported license

Blockchain Technology for Collaborative Information Systems, *Dagstuhl Reports*, Vol. 8, Issue 08, pp. 67–129

Editors: Marlon Dumas, Richard Hull, Jan Mendling, and Ingo Weber



DAGSTUHL  
REPORTS

Dagstuhl Reports  
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

control, thus allowing multiple parties to selectively share their data with each other and to selectively grant permissions to perform transactions on these data.

While blockchain opens up new possibilities, it also raises a number of challenges because it requires us to re-think the way B2B collaborations are designed and implemented. In contrast to centralized collaborative processes, the transparent and decentralized nature of blockchains brings in new challenges related to compliance, control, and privacy, in addition to major scalability and performance challenges. This seminar brought together established and young researchers with forward-thinking industry representatives from both large and start-up companies, in order to establish a research roadmap for blockchain-based collaborative information systems, and to initiate concrete research collaborations between participants along this roadmap.

## 2 Table of Contents

### Executive summary

<i>Marlon Dumas, Richard Hull, Jan Mendling, and Ingo Weber</i> . . . . .	67
---	----

### Overview of Talks

Collaborative Business Process Execution on Blockchain: The Caterpillar System <i>Marlon Dumas and Luciano García-Bañuelos</i> . . . . .	71
Shared Ledger Business Collaboration Language <i>Richard Hull</i> . . . . .	71
Assessing the impact of emerging information technologies on processes <i>Jan Mendling</i> . . . . .	72
Software Architecture and Engineering for Blockchain Applications <i>Ingo Weber</i> . . . . .	72
Research for the engineering of blockchain-based systems <i>Mark Staples</i> . . . . .	73
HyperPubSub: a Decentralized, Permissioned, Publish/Subscribe Service using Blockchains <i>Kaiwen Zhang, Hans-Arno Jacobsen, and Nejc Zupan</i> . . . . .	74
Tracking Business Processes on the Blockchain <i>Claudio Di Ciccio</i> . . . . .	74
Lorikeet: A Model-Driven Engineering Tool for Blockchain-Based Business Process Execution and Asset Management <i>Qinghua Lu</i> . . . . .	75
Healthcare Data Management Using Blockchain: Open Challenges and Lessons Learned <i>Alevtina Dubovitskaya</i> . . . . .	75
Obsidian: A Safer Blockchain Programming Language <i>Michael Coblenz</i> . . . . .	76
Ethereum-Based execution of DMN decisions <i>Stephan Haarman</i> . . . . .	76
Collaboration among Adversaries: Distributed Declarative Workflow Execution on a Blockchain <i>Søren Debois</i> . . . . .	77
Ergo: A Strongly Typed DSL for Smart Legal Contracts <i>Jerome Simeon</i> . . . . .	77
Introduction to Hyperledger Fabrics <i>Petr Novotny</i> . . . . .	77
Blockchain Research: Process Verification and Beyond <i>Stefan Schulte</i> . . . . .	78

### Working Groups

Bridging the Gap between IoT and Blockchain: Research Questions & Challenges <i>Fabiana Fournier, Agnes Koschmider, Raimundas Matulevičius, Sooyong Park,</i> <i>Stefan Schulte</i> . . . . .	78
---	----

Two Perspectives on Blockchains: Capabilities vs. Features <i>Søren Debois, Marlon Dumas, Stephan Haarmann, Hans-Arno Jacobsen, Mieke Jans, Jan Mendling, Mark Staples, Barbara Weber, Francesca Zerbato, Kaiwen Zhang</i>	82
Factors Influencing Process Analytics on Distributed Ledgers <i>Claudio Di Ciccio, Luciano García-Bañuelos, Mieke Jans, Jan Mendling, Petr Novotny, Ludwig Stage</i>	86
A Holistic Vision of Blockchain-Based Application Design, Specification, and Implementation <i>Michael Coblenz, Richard Hull, Qinghua Lu, Ingo Weber</i>	90
Towards a Blockchain Collaboration Meta-Model and Language <i>Michael Coblenz, Claudio Di Ciccio, Marlon Dumas, Fabiana Fournier, Luciano García-Bañuelos, Richard Hull, Qinghua Lu, Raimundas Matulevičius, Jérôme Siméon, Mark Staples, Ingo Weber</i>	94
Blockchain Data Analytics: Example of Decentralization of Service-Provider Platform <i>Alevtina Dubovitskaya, Avigdor Gal, Stephan Haarmann, Stefanie Rinderle-Ma, Francesca Zerbato</i>	107
Data Technology to the Rescue: Digging the <b>D</b> in GDPR <i>Søren Debois, Alevtina Dubovitskaya, Avigdor Gal, Petr Novotny, Stefanie Rinderle-Ma, Stefan Schulte, Ludwig Stage, Kaiwen Zhang</i>	115
<b>Participants</b>	129

### 3 Overview of Talks

This section provides an overview of all the talks held by participants during the seminar.

#### 3.1 Collaborative Business Process Execution on Blockchain: The Caterpillar System

*Marlon Dumas (University of Tartu, EE) and Luciano García-Bañuelos (University of Tartu, EE)*

**License** © Creative Commons BY 3.0 Unported license  
 © Marlon Dumas and Luciano García-Bañuelos  
**Joint work of** Orlenys López-Pintado, Luciano García-Bañuelos, Marlon Dumas, Ingo Weber, Alexander Ponomarev  
**Main reference** Orlenys López-Pintado, Luciano García-Bañuelos, Marlon Dumas, Ingo Weber, Alexander Ponomarev: “CATERPILLAR: A Business Process Execution Engine on the Ethereum Blockchain”, CoRR, Vol. abs/1808.03517, 2018.  
**URL** <https://arxiv.org/abs/1808.03517>

Blockchain platforms allow a set of actors to maintain a ledger of transactions without relying on a central authority and to deploy scripts, called smart contracts, that are executed whenever certain transactions occur. These features can be used as building blocks to support the execution of collaborative business processes between mutually untrusting parties. However, implementing business processes using the low-level primitives provided by blockchain platforms is cumbersome and error-prone. In contrast, established business process management systems, such as those based on the standard Business Process Model and Notation (BPMN), provide convenient abstractions for rapid development of process-oriented applications. In this talk, we show how to combine the advantages of a business process management system with those of a blockchain platform. Specifically, we present a BPMN execution engine, namely Caterpillar, designed to support collaborative business processes on top of a blockchain platform. Like any BPMN execution engine, Caterpillar supports the creation of instances of a process model and allows users to monitor the state of process instances and to execute tasks thereof. The specificity of Caterpillar is that the state of each process instance is maintained on the (Ethereum) blockchain and the workflow routing is performed by smart contracts generated by a BPMN-to-Solidity compiler. The Caterpillar compiler supports a large array of BPMN constructs, including subprocesses, multi-instances activities and event handlers.

#### 3.2 Shared Ledger Business Collaboration Language

*Rick Hull (IBM TJ Watson Research Center – Yorktown Heights, US)*

**License** © Creative Commons BY 3.0 Unported license  
 © Richard Hull

Blockchain offers the possibility of a fundamentally new way to use information processing in support of business collaboration. We may see a transition from collaboration based on families of binary relationships managed through messaging, to collaboration based on holistic groups of organizations guided by a single “source-of-truth” data repository and shared processing logic. Current approaches to business process and operations management will be extended and transformed as the possibilities and benefits of the single source of

truth are understood and leveraged. This will bring opportunities to re-think current process models and best practices for reducing the models to executables. It also raises challenges in the area of “on-boarding” companies into the usage of Blockchain, because techniques are needed to enable smooth integration between Blockchain-hosted processing and the legacy processing that remains off of Blockchain.

### 3.3 Assessing the impact of emerging information technologies on processes

*Jan Mendling (Wirtschaftsuniversität Wien, AT)*

**License** © Creative Commons BY 3.0 Unported license  
© Jan Mendling

**Joint work of** Jan Mendling, Gero Decker, Richard Hull, Hajo Reijers, Ingo Weber

**Main reference** Mendling, Jan and Decker, Gero and Richard, Hull and Hajo A., Reijers and Ingo, Weber: “How do Machine Learning, Robotic Process Automation, and Blockchains Affect the Human Factor in Business Process Management?” *Communications of the Association for Information Systems*, 43 (Art.19). pp. 297–320, 2018.

**URL** <https://aisel.aisnet.org/cais/vol43/iss1/19/>

This talk discusses the impact of emerging technologies on the way how processes can be executed. It makes the point that the often asked naive question “can we do this on the blockchain?” is misleading. Those blockchain variants that come with a Turing-complete programming language can capture any representation of states and transitions.

#### References

- 1 Jan Mendling, et al.: Blockchains for business process management-challenges and opportunities. *ACM Transactions on Management Information Systems (TMIS)* 9.1 (2018): 4.

### 3.4 Software Architecture and Engineering for Blockchain Applications

*Ingo Weber (Data61, CSIRO – Sydney, AU)*

**License** © Creative Commons BY 3.0 Unported license  
© Ingo Weber

**Main reference** Xiwei Xu, Ingo Weber, and Mark Staples. *Architecture for Blockchain Applications*. Springer, 2019.

Blockchain technology is emerging and impactful, but its functional and non-functional influence on software applications was initially not well understood. My team and I have been addressing this problem over the past three years, and published a number of papers on various aspects of it. In this talk, I will give an overview over some of our work. As such, I will first present the terminological definitions of blockchain as a concept, technology, network, etc. from our forthcoming book. Then I will provide an overview of the roles blockchain can play in software applications, and what its non-functional properties are. Subsequently I will discuss our design process for choosing a blockchain and configuration. This process starts with the assessment of suitability of blockchain technologies for a given context. Finally, I will provide a short overview of our blockchain design patterns collection. Topics of model-driven engineering and our empirical work are covered in other talks at this seminar, and will therefore not be part of my talk.

### 3.5 Research for the engineering of blockchain-based systems

Mark Staples (*Data61, CSIRO – Sydney, AU*)

License © Creative Commons BY 3.0 Unported license  
© Mark Staples

We will increasingly rely on blockchains (including distributed ledger technologies more broadly) for critical services. Blockchains have the potential to create, and thus put at risk, significant business value. Increasingly blockchains are also being planned to be used for safety-critical applications such as for health and Internet of Things (IoT). So it is important for researchers to not only understand how to design blockchain-based systems, but also to create evidence that they will function correctly, so we can use trustworthy blockchain-based systems.

Functionally, blockchains are a kind of database (an append-only ledger) and a compute platform (executing smart contracts). However, conventional database and compute platforms are owned, operated, and/or administered by single organisations, who become a single point of technical or business failure for the platform. Blockchains are instead operated by a collective. For public blockchains the collective is a large group of anonymous contributors, and in a consortium blockchain the collective is usually defined by contractual arrangements between organisations. Blockchains support efficient and trustworthy ways for organisations to work together. Blockchains also support the representation and control of digital assets, which (unlike normal information assets) can be transferred as forms of exclusive property held by individual parties. Blockchains are potentially disruptive because for the basic services supporting transactional business relationships in industry and society, we can now choose to rely on the neutral territory provided by blockchains, instead of relying on trusted third-parties to facilitate those relationships.

Blockchains have non-functional differences to conventional databases and compute platforms, which impact the architectural design of blockchain-based systems. For example, blockchains have very strong support for integrity, but struggle to directly support confidentiality. Good architectural design is critical to allow systems to benefit from the strengths of blockchain platform, and shore up their weak areas using other off-chain components.

I describe some of the blockchain research from Data61 (CSIRO) that addresses these challenges. This includes the Lorikeet model-driven development platform for generation of blockchain-based systems for business process execution and monitoring, and for control of data-centric business objects in registries. The back-end targets for system generation include Ethereum and Hyperledger Fabric. Our research has also explored the use of these process models for simulation-based performance prediction and for cost analysis. This model-driven generation of systems can take advantage of experience in blockchain architectural design which we have begun to capture in blockchain design patterns. Data61's other blockchain research includes work towards the mechanised formal verification of smart contracts on Ethereum's Virtual Machine, and work towards using declarative representations of legal contracts, with the vision to be used either as specifications for smart contracts, or else as smart contracts, directly interpreted on blockchain platform infrastructure. In other research, we have begun to explore frameworks to integrate support for GS1's EPCIS event data standard into blockchain applications. Although most of our research is on the level of blockchain-based systems, we also have some research activities on the level of underlying blockchain platform, through the development of high-throughput, low-latency consensus mechanisms with a conventional transaction commit semantics, realised in the Red Belly Blockchain. Other platform-level research investigates the use of blockchain principles for IoT networks.

### 3.6 HyperPubSub: a Decentralized, Permissioned, Publish/Subscribe Service using Blockchains

*Kaiwen Zhang (ETS – Montreal, CA), Hans-Arno Jacobsen (TU München, DE), and Nejc Zupan*

**License** © Creative Commons BY 3.0 Unported license

© Kaiwen Zhang, Hans-Arno Jacobsen, and Nejc Zupan

**Joint work of** Nejc Zupan, Kaiwen Zhang, Hans-Arno Jacobsen

**Main reference** Nejc Zupan, Kaiwen Zhang, Hans-Arno Jacobsen: “Hyperpubsub: a decentralized, permissioned, publish/subscribe service using blockchains: demo”, in Proc. of the 18th ACM/IFIP/USENIX Middleware Conference: Posters and Demos, Las Vegas, NV, USA, December 11–15, 2017, pp. 15–16, ACM, 2017.

**URL** <https://doi.org/10.1145/3155016.3155018>

Since the introduction of Bitcoin in 2008, blockchain systems have evolved immensely in terms of performance and usability. There is a massive focus on building enterprise blockchain solutions, with providers such as IBM and Microsoft already providing Blockchain-as-a Service (BaaS). To facilitate the adoption of blockchain technologies across various business verticals, we argue that middleware plays an integral role in accelerating the development of automated business processes (i.e., smart contracts). We argue that decentralized messaging is a key requirement of many distributed applications and should be provided as a reusable blockchain middleware. Our system, called HyperPubSub, provides decentralized publish/subscribe messaging for a multi-federated, permissioned, environment. HyperPubSub provides secure and privacy-preserving messaging, which is audited using blockchains for validation and monetization purposes. We demonstrate our implementation using Kafka and Hyperledger.

### 3.7 Tracking Business Processes on the Blockchain

*Claudio Di Ciccio (Wirtschaftsuniversität Wien, AT)*

**License** © Creative Commons BY 3.0 Unported license

© Claudio Di Ciccio

Blockchain technology opens up new opportunities for Business Process Management. This is mainly due to its unprecedented capability to let transactions be automatically executed and recorded by Smart Contracts in multi-peer environments, in a decentralised fashion and without central authoritative players to govern the workflow. In this way, blockchains also provide traceability. Traceability of information plays a pivotal role particularly in those supply chains where multiple parties are involved and rigorous criteria must be fulfilled to lead to a successful outcome. In this talk, we investigate how to run a business process in the context of a supply chain on a blockchain infrastructure so as to provide full traceability of its run-time enactment. Our approach retrieves information to track process instances execution solely from the transactions written on-chain. To do so, hash-codes are reverse-engineered based on the Solidity Smart Contracts’ encoding of the generating process. We show the results of our investigation by means of an implemented software prototype, with a case study on the reportedly challenging context of the pharmaceutical supply chain.



### 3.8 Lorikeet: A Model-Driven Engineering Tool for Blockchain-Based Business Process Execution and Asset Management

*Qinghua Lu (Data61, CSIRO – Sydney, AU)*

**License** © Creative Commons BY 3.0 Unported license  
© Qinghua Lu

**Joint work of** An Binh Tran, Qinghua Lu, Ingo Weber

**Main reference** An Binh Tran, Qinghua Lu, Ingo Weber: “Lorikeet: A Model-Driven Engineering Tool for Blockchain-Based Business Process Execution and Asset Management”, in Proc. of the Dissertation Award, Demonstration, and Industrial Track at BPM 2018 co-located with 16th International Conference on Business Process Management (BPM 2018), Sydney, Australia, September 9-14, 2018., CEUR Workshop Proceedings, Vol. 2196, pp. 56-60, CEUR-WS.org, 2018.

**URL** [http://ceur-ws.org/Vol-2196/BPM\\_2018\\_paper\\_12.pdf](http://ceur-ws.org/Vol-2196/BPM_2018_paper_12.pdf)

Blockchain has attracted a broad range of interests including startups, enterprises and governments. A large amount of projects have been conducted to explore how to use blockchain to re-architect systems and to build new applications and business models. However, blockchain is a new technology with still limited tooling and documentation, so there can be a steep learning curve for developers. In addition, it is difficult to fix bugs by releasing new versions of smart contracts and mistakes in smart contracts have led to massive economic loss, such as the DAO exploit. Thus, in this seminar, we demonstrate a model-driven development tool, named Lorikeet, which can automatically generate smart contracts from business process models and/or registry data schema. The architecture of Lorikeet, which consists of a BPMN and registry modeller, smart contract generator and blockchain trigger. The BPMN and registry modeller is presented as a web application for users to build business process and registry models. Business processes are modelled in the Business Process Model and Notation (BPMN) 2.0, while registries are modelled as XML schema. The smart contract generator consists of the BPMN translator and the registry generator. The BPMN translator can automatically create smart contracts in Solidity from BPMN models while the registry generator can produce smart contract based on the registry models. The blockchain trigger communicates with a blockchain node and handles compilation, deployment and interactions with smart contracts. Lorikeet is a well-tested development tool that is used for producing blockchain smart contracts in industry and academia.

### 3.9 Healthcare Data Management Using Blockchain: Open Challenges and Lessons Learned

*Alevtina Dubovitskaya (EPFL – Lausanne, CH)*

**License** © Creative Commons BY 3.0 Unported license  
© Alevtina Dubovitskaya

**Joint work of** Alevtina Dubovitskaya, Zhigang Xu, Samuel Ryu, Michael Schumacher, Fusheng Wang

**Main reference** Alevtina Dubovitskaya, Zhigang Xu, Samuel Ryu, Micheal Schumacher, and Fusheng Wang. “Secure and Trustable Electronic Medical Records Sharing using Blockchain.” AMIA Annual Symposium Proceedings, Washington, DC, USA, November 4–8, 2017, 2017:650–659, 2018.

Healthcare data are sensitive but have to be frequently shared among the peers in the healthcare network. Blockchain is a distributed ledger technology that may execute arbitrary, programmable transaction logic in the form of smart contracts, and provides a shared, immutable, and transparent append-only register of all the actions that have happened in the network. This provides a unique opportunity to employ blockchain technology to facilitate and enhance healthcare data management for all the actors in the healthcare network. The

immense development of the blockchain technology is happening over the last years, and the interest to the potential of its application in healthcare is growing. However, a quest for such system that guarantees privacy, security, scalability, and efficiency continues. In this talk we discuss potential applications of blockchain in health and present a prototype for blockchain-based consent-management system. We discuss the challenges of adopting such system in medical practice.

### 3.10 Obsidian: A Safer Blockchain Programming Language

*Michael Coblentz (Carnegie Mellon University – Pittsburgh, US)*

License  Creative Commons BY 3.0 Unported license  
© Michael Coblentz

Blockchain programs have been repeatedly exploited by hackers due to security vulnerabilities, or otherwise found incorrect or unsafe. Verification is one approach toward correct code, but so far, verification is an impractical technique for most programmers. In contrast, Obsidian is a new language that provides particular safety guarantees via an expressive, strong type system. Obsidian allows users to lift state information into types so that the compiler can enforce certain kinds of protocol adherence. Obsidian also supports reasoning about resources, expressing these with linear types.

Obsidian is designed in a user-centered way and serves as a testbed for user-centered language design techniques. We incorporate formative HCI methods to make it more likely that the language is as effective as possible for programmers. We plan to use summative HCI methods in the hope of showing that Obsidian is a more effective tool than prior languages for users to write blockchain programs.

### 3.11 Ethereum-Based execution of DMN decisions

*Stephan Haarman (Hasso-Plattner-Institut – Potsdam, DE)*

License  Creative Commons BY 3.0 Unported license  
© Stephan Haarman

In business collaborations, participants interact and cooperate to reach business goals. Therefore, their business processes are interconnected via message exchange. Successful collaboration, however, does not only depend on the exchange of messages. Furthermore, local processes must comply with constraints and rules. These are traditionally given by a contract, but conflicts (such as a misunderstanding of the terms) are detected lately, and resolving them becomes expensive. Blockchain technology can help by providing a single source of truth for the data and executable, unambiguous terms (logic).

Using the Decision Model and Notation standard, stakeholders can model business rules precisely. This demo presents a compiler that translates a DMN decision model into smart contract code. The smart contract can be deployed on the Ethereum blockchain: it represents an agreement between participants. An instant specific state (contract) is a single source of truth for all the data. Consequently, all participants can rely on a shared version of data and logic. If all participants comply, then the shared information prevents conflicts. If a conflict occurs, the on-chain data helps to resolve it.

### 3.12 Collaboration among Adversaries: Distributed Declarative Workflow Execution on a Blockchain

*Søren Debois (IT University of Copenhagen, DK)*

**License** © Creative Commons BY 3.0 Unported license  
© Søren Debois

We study distributed declarative workflow execution in an adversarial setting. In this setting, parties to an agreed-upon workflow do not trust each other to follow that workflow, or suspect the other party might misrepresent proceedings at a later time. We demonstrate how distributed declarative workflow execution can be implemented as smart contracts, guaranteeing (I) enforcement of workflow semantics, and (II) an incontrovertible record of workflow execution history. Crucially, we achieve both properties without relying on a trusted third party. The implementation is based on the Ethereum blockchain, inheriting the security properties (I) and (II) from the guarantees given by that chain. A recurring challenge for both the implementation and the analysis is the cost of operations on Ethereum: This cost must be minimised for honest parties, and an adversary must be prevented from inflicting extra cost on others

### 3.13 Ergo: A Strongly Typed DSL for Smart Legal Contracts

*Jerome Simeon (Clause Inc. – New York, US)*

**License** © Creative Commons BY 3.0 Unported license  
© Jerome Simeon

Smart contracts are not legal unless they are legal contracts. Legal contracts are not smart unless they are made computable. Smart legal contracts should not only be legal and smart, but also safe and portable. Ergo is a new DSL which aims to have all those properties. It is developed as part of the Accord Project consortium and open source. It is strongly typed and has a reference implementation and compiler written using the Coq proof assistant.

### 3.14 Introduction to Hyperledger Fabrics

*Petr Novotny (IBM TJ Watson Research Center – Yorktown Heights, US)*

**License** © Creative Commons BY 3.0 Unported license  
© Petr Novotny

**Joint work of** Angel Nunez Mencias, Donna Dillenberger, D; Petr Novotny, Fabian Toth, Thomas E. Morris  
**Main reference** Angel Nuñez Mencias, Donna N. Dillenberger, Petr Novotny, Fabian Toth, Thomas E. Morris, Volodymyr Paprotski, John C. Dayka, Tamas Visegrady, Bill O’Farrell, Jakob Lang, Ellen Carbarnes: “An optimized blockchain solution for the IBM z14”, IBM Journal of Research and Development, Vol. 62(2/3), p. 4, 2018.

**URL** <https://doi.org/10.1147/JRD.2018.2795889>

Hyperledger is an open source project hosted by Linux Foundation, supporting collaborative development of blockchain technologies. IBM is the key contributor to the Hyperledger Fabric and to other Hyperledger projects. In this talk, we will explore the key features of permissioned blockchain platform Hyperledger Fabric and of other Hyperledger projects, with the focus on the transaction processing and consensus architecture and protocols, and on the privacy and access control mechanisms. We will then look at a number of use cases which

leverage the permissioned blockchain features in solutions for supply chain, international trade, finance, and others. Finally, we will explore IBM technologies designed for rapid development, simple operation and high performance of blockchain networks.

### 3.15 Blockchain Research: Process Verification and Beyond

*Stefan Schulte (TU Wien, AT)*

**License** © Creative Commons BY 3.0 Unported license  
© Stefan Schulte

**Joint work of** Christoph Prybila, Stefan Schulte, Christoph Hochreiner, Ingo Weber  
**Main reference** Christoph Prybila, Stefan Schulte, Christoph Hochreiner, and Ingo Weber: “Runtime Verification for Business Processes Utilizing the Bitcoin Blockchain.” *Future Generation Computer Systems*, 2018.

**URL** <https://doi.org/10.1016/j.future.2017.08.024>

The documentation and verification of real-world events plays an important role in smart systems, e.g., with regard to supply chains or logistics. Once events have been identified, it is necessary to distribute them to data stakeholders using a trusted channel. Especially in distributed scenarios, where different organizations might be involved, it is necessary to store this data in an immutable way.

In the world of cryptocurrencies like Bitcoin, a similar problem arises, since transactions need to be stored in a permanent and unchangeable way without relying on a trusted third party. For this, the Blockchain is applied. Within this talk, we will outline how Blockchains can be used in order to provide runtime documentation and verification in business processes. In particular, we will present how the Bitcoin Blockchain is used in a concrete solution to achieve runtime verification for distributed, choreography-based business process execution.

## 4 Working Groups

In this section, we summarize the main results achieved by selected working groups.

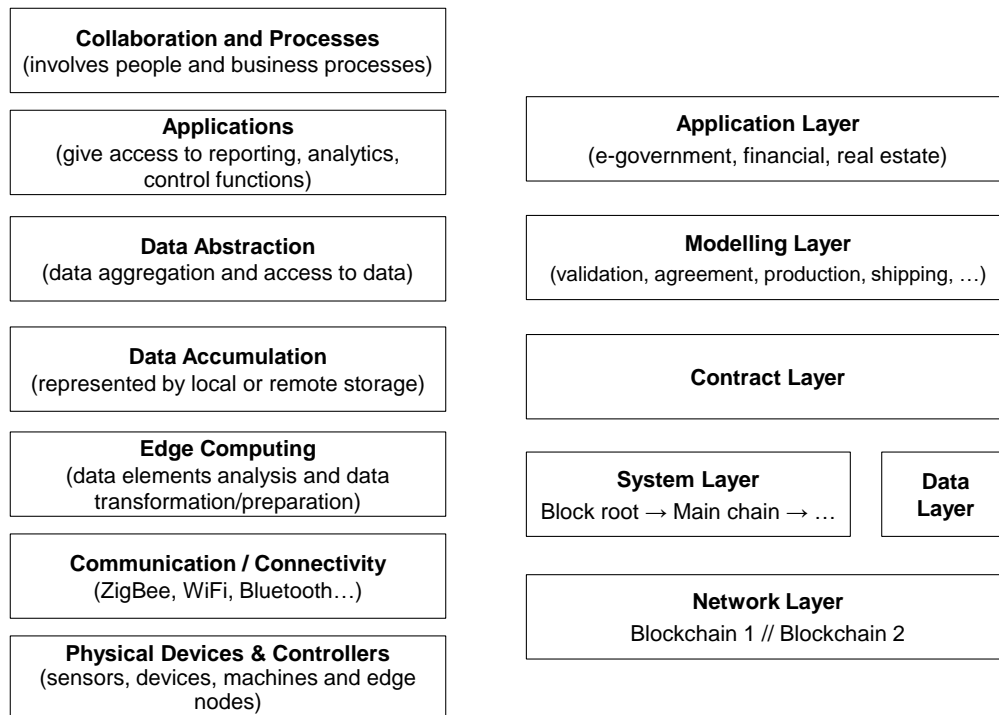
### 4.1 Bridging the Gap between IoT and Blockchain: Research Questions & Challenges

*Fabiana Fournier (IBM – Haifa, IL), Agnes Koschmider (KIT – Karlsruher Institut für Technologie, DE), Raimundas Matulevičius (University of Tartu, EE), Sooyong Park (Sogang University – Seoul, KR), and Stefan Schulte (TU Wien, AT)*

**License** © Creative Commons BY 3.0 Unported license  
© Fabiana Fournier, Agnes Koschmider, Raimundas Matulevičius, Sooyong Park, Stefan Schulte

In short, the Internet of Things (IoT) is defined as the pervasion of business and private spaces with “a variety of things or objects . . . [which] interact with each other and cooperate with their neighbors to reach common goals” [3], forming “an interconnected world-wide network based on sensory, communication, networking, and information processing technologies” [35].

Blockchain technologies have been named as a facilitator for the IoT with regard to a number of different issues [15, 53], e.g., the verification of IoT data and actions.



■ **Figure 1** Layer Models for IoT (Left) and Blockchain (Right) [16, 70].

To categorize the application potential of blockchain technologies in the IoT and to identify research challenges, we make use of the IoT reference model introduced by Cisco [16], and the blockchain layer model introduced by Zhang and Jacobsen [70]. Figure 1 gives an overview of these two layer-based models.

In the following, we will discuss a number of particular research challenges with regard to the IoT and blockchain technologies. These comprise challenges where blockchain technologies may help to overcome IoT-related issues, as well as research questions which stem from the fact that IoT technologies and standard blockchains have partially contradicting features. For each research challenge, we describe where it is located in the combination of the two layer models.

#### 4.1.1 Research Challenges

**Storage capacity and latency:** IoT nodes are generating data in high volume, high velocity, high variety (also known as the original “3 Vs” in Big Data [14]), and low veracity [29]. This is contradictory to some essential characteristics of blockchains: First, smart systems consisting of many IoT nodes may generate huge amounts of data, while data storage in blockchains is usually expensive, leading to the need to find ways to pre-filter data and/or store (parts of the) data off-chain in order to decrease the amount of data which needs to be stored on-chain. Second, the velocity of IoT data makes it necessary to process it fast (especially in data stream processing [27]), while the generation of blocks in blockchains leads to an inherent delay. Therefore, blockchain technologies may be used for other purposes, but not necessarily for real-time data exchange in the IoT. Third,

data variety makes it necessary to provide solutions that different data types can be used in blockchain-based systems. Finally, IoT data may lead to issues regarding data veracity, making it necessary to change stored data. However, blockchains rely on the principle that data cannot be changed. Hence, new solutions, e.g., for data versioning in the blockchain, need to be provided.

With regard to Figure 1, these research challenges can be attributed to both the *Data Abstraction* and *Communication / Connectivity* layers in the IoT reference model, and the *Data Layer* and *Network Layer* in the blockchain layer model.

**Scalability:** IoT-based smart systems can scale up and down very quickly, i.e., the number of nodes (and therefore the amount of data) in the IoT may vary a lot over time. However, some blockchain technologies, e.g., the Bitcoin blockchain, have trouble with scalability, since they are restricted to a certain amount of transactions per time frame. Therefore, it is important to select a sufficient blockchain technology for IoT scenarios. In some cases, the usage of side-chains might be a good advise.

With regard to Figure 1, this research challenge can be attributed to the *System Layer* in the blockchain layer model.

**Security:** Security is an issue on all levels of the IoT stack. Blockchains may be able to help to take care of some of the issues here, e.g., to identify nodes and entities in general, based on the keys already used in blockchains. Permissioned blockchains may be used to establish closed IoT systems, where data is only shared and forwarded between known partners and nodes, thus decreasing security issues.

With regard to Figure 1, this is an overarching research challenge, which needs to be regarded on all layers in both the IoT reference model and the blockchain layer model.

**Anonymity and data privacy:** Per se, data which is stored on a blockchain is available to all parties which can access the blockchain. Even though parties are anonymized (or rather pseudonymized) through their blockchain addresses (as in Bitcoin), their identities might be derived from the stored data. Therefore, data models are necessary which can be used in order to categorize which kind of data can be stored in which kind of blockchain, or which data should only be stored offline.

With regard to Figure 1, this is an overarching research challenge, which needs to be regarded on all layers in both the IoT reference model and the blockchain layer model. However, naturally, the *Data Layer* in the blockchain layer model and the *Data Abstraction* and *Data Accumulation* layers in the IoT reference model are of primary interest.

**Smart contracts:** With regard to IoT technologies, there are some examples for simple Event-Condition-Action smart contracts, e.g., to release funds once an (IoT-equipped) shipment has been delivered, e.g., [34], but there may be potential for further types of smart contracts in the IoT. In addition, data velocity may play a role here: If an oracle in a smart contract is based on IoT data, which is outdated very fast, this may lead to inconsistencies and conflicts, so that the execution of a smart contract cannot be validated. Some applications of IoT require timing constraints (e.g., fire alarm starts within 0.1 sec if temperature over 80°C in a smart farm). If this situation is controlled by a smart contract, there is a need for a real-time smart contract that satisfies timing constraints.

With regard to Figure 1, this is a research challenge closely related to the *Contract Layer* in the blockchain layer model.

**Legal issues:** While not IoT-specific, the legal status of smart contracts and blockchain-hosted data is not clear in all countries. In general, IoT data underlies data privacy legislation, as has been discussed above. Hence, data privacy legislation like the EU's

General Data Protection Regulation needs to be taken into account when storing data in an append-only store like the blockchain.

While not a legal issue, blockchains may also help to overcome legal disputes, by providing data documenting or verifying a particular situation (e.g., in autonomous driving [19]).

With regard to Figure 1, this research challenge can be attributed to the *Applications* layer in the IoT reference model, and the *Application Layer* in the blockchain layer model.

**Consensus:** While there might be powerful edge devices in the IoT, a very large percentage of all IoT devices only possess minor amounts of computational resources. Hence, mining and taking part in consensus finding on these devices is very difficult to achieve, especially in Proof of Work (PoW)-based blockchains. This makes it necessary to come up with solutions where IoT devices use a proxy to participate in a blockchain network, or are decoupled from a blockchain in another way. Also, blockchains which do not make use of PoW could be beneficial in IoT scenarios, since it might not be necessary to find consensus (but to validate new blocks) in the first place. Also, more lightweight consensus algorithms could be applied in the first place.

With regard to Figure 1, this research challenge can be attributed to the *System Layer* in the blockchain layer model and the *Physical Devices & Controllers* layer in the IoT reference model.


**Low computing power and network issues:** As already pointed out above, IoT devices may not be powerful enough to participate in certain blockchain-related tasks. This calls for lightweight consensus algorithms, abstract block mechanisms, lightweight hash functions, and also a minimum communication protocol.

IoT devices might be battery-powered, which makes it necessary to run them in an energy-efficient way. Thus, such devices cannot be “always-on”. Also, data transfer between a blockchain-based backbone and the IoT devices needs to be limited – not only because of the energy consumption in the case of data transmissions, but also, since the network might actually become a bottleneck in large-scale IoT-based systems, where a huge number of entities are sending and receiving data. Together with the already mentioned consensus-related issues, this calls for blockchain solutions where IoT devices only join the network in particular cases, and are more or less dormant at all other times. This avoids that the blockchain-inherent communication overhead leads to rapid energy consumption.

With regard to Figure 1, this research challenge can be attributed to the *System Layer* in the blockchain layer model and the *Physical Devices & Controllers* and *Communication/Connectivity* layers in the IoT reference model.

## 4.2 Two Perspectives on Blockchains: Capabilities vs. Features

*Søren Debois (IT University of Copenhagen, DK), Marlon Dumas (University of Tartu, EE), Stephan Haarmann (Hasso Plattner Institut, DE), Hans-Arno Jacobsen (TU München, DE), Mieke Jans (Hasselt University, BE), Jan Mendling (Wirtschaftsuniversität Wien, AT), Mark Staples (Data61, CSIRO –Sydney, AU), Barbara Weber (Technical University of Denmark – Lyngby, DK), Francesca Zerbato (University of Verona, IT), and Kaiwen Zhang (ETS – Montreal, CA)*

**License**  Creative Commons BY 3.0 Unported license  
 © Søren Debois, Marlon Dumas, Stephan Haarmann, Hans-Arno Jacobsen, Mieke Jans, Jan Mendling, Mark Staples, Barbara Weber, Francesca Zerbato, Kaiwen Zhang

Blockchain technology is the subject of substantial enthusiasm and notable financial successes. For example in June 2018 alone, almost USD\$6B worth of tokens were issued in ICOs <sup>1</sup>.

Indications of widespread use of blockchain and distributed ledger technologies outside of tokens and cryptocurrency are emerging.

Prior work proposed different decision models with the goal to help answering the question “Do i need a blockchain for my application?” [21, 26, 39, 41, 42, 44, 50, 67, 68, 71]. Moreover, there are proposals to guide the design process (i.e., which blockchain configuration to best choose) (e.g., [68]). However, there is little work up to now that focuses on the business capabilities that might form part of a blockchain-based application supporting business operations and on how they link to blockchain features.

In the remainder, we proceed as follows. In Sect. 4.2.1 we provide the foundations of blockchain and distributed ledger technologies. Then, in Sect. 4.2.3 we give a list of business capabilities identified as key to blockchain. Moreover, in Sect. 4.2.4 we identify a list of blockchain features. Subsequently, we map features to blockchain capabilities. Finally, we outline potential future work in Sect. 4.2.5.

### 4.2.1 Background

In this section, we recall the main concepts of blockchain and distributed ledger technologies.

#### Blockchain and Distributed Ledger Technologies

Applications of blockchain typically shift trust from a third party (a bank, a government institution, a credit card company) onto something else, typically the technology of the chain itself. There are two reasons one might desire such a shift:

1. One does not wish to trust the third party. (Bitcoin: government-less money)
2. The third party is expensive. (Hypothetical example: Credit card companies.)

However, new applications might arise where there previously were no solution because involved parties could not or would not agree on a trusted third party. For example, Mærsk and other shipping companies always had the option of developing a global, centralised repository of shipping documents; however, presumably, who would control that repository prevented it from coming into existence.

We emphasise that in the absence of risk or trust issues, a blockchain has no purpose. In other words, *a blockchain is needed only if the data consumers and the data owner are in separate trust domains and the consumer has high-integrity requirements. There is no need*

---

<sup>1</sup> <https://www.coinschedule.com/stats.html>



for a blockchain when the data consumer(s) and data owner are in the same trust domain (e.g. inside a company).

To understand what capabilities are central to / indicative of such shifting of trust, we first (attempt) to clarify what is “trust” and what is “a capability”.

**Definition of trust.** Trust is the acceptance of risk. Such risk may arise either from, say, malicious intent, or unintentional byzantine errors (either because of incompetence or because of hostile environment)

#### **Alternative viewpoint (solution-driven).**

The benefits conferred from blockchain technology constitutes “affordances” (see: Gibson, J.J., *The Ecological Approach to Visual Perception*. 1979. Boston: Houghton-Mifflin) rather than a outright features:

- Having trust in a system without having a trusted third party;
- Lower cost for the service;
- Lower barrier of entry;
- More accessible than traditional services (strategic advantage);
- Elimination of TTP;
- Tolerance to failures (impact of failures).

#### **4.2.2 Capabilities and the Resource-Based View of the Firm**

Business each have a wide range of capabilities<sup>2</sup>. Some are strategic capabilities which are key to the business’s sustainable competitive advantage, and are valuable and distinctive compared to other businesses. Strategic capabilities are sometimes called “core competencies”. Others are operational capabilities, which are necessary for the operation of the business, but will not be distinctive, and are more likely to be outsourced. A capability area may be strategic for one business, but operational for another.

Blockchains provide a mechanism allowing businesses to shift trust within the operation of their ecosystems. Often this is for disintermediation, stopping the centralised control of that capability by those third parties. This can be good for businesses that want to use that capability as an operational capability. However for trusted third-parties, this capability is a strategic capability, and blockchain may directly undermine the sustainability of their competitive advantage from that capability.

**Definition of capability.** “Capability thinking also means being aware of in what context the enterprise has the capacity and ability to offer business services that contribute to achieving business goals. The context basically captures what legal, technical, process, content, or other situation the business service is prepared for and what variations in providing the business service apply for what situation” [54].

#### **4.2.3 Capabilities of Blockchain-Based Systems**

Table 1 shows the main business capabilities resulting from our analysis and discussion. The list of business capabilities we have identified below is not exhaustive, and the capabilities

<sup>2</sup> Here we do not mean “object capabilities” which are secure references used in capability-based security models. We also do not mean software engineering capabilities captured for example in models such as CMMI.

■ **Table 1** Business capabilities for blockchain-based systems.

BC1: Voting <ul style="list-style-type: none"> <li>■ Anonymous voting</li> <li>■ Delegatable voting (conditional voting with smart contracts)</li> <li>■ Number of participants(N): un/bounded</li> <li>■ Non-sellable</li> </ul>	Entry of votes submitted by different parties, tallying, and announcement of results.
BC2: Payment <ul style="list-style-type: none"> <li>■ Anonymous payments</li> <li>■ Escrow payments</li> <li>■ Variable payments</li> <li>■ Complex conditional payment</li> </ul>	Transfer of cryptocurrency between different parties.
BC3: Asset transfer	The transfer of assets (cryptocurrency, tokens) from one party to another.
BC4: Settlement (payment vs. delivery)	Synchronisation of simultaneous asset transfers.
BC5: Exchanges	Settlement of particular assets.
BC6: Introductions	Connecting parties interested in being end-points of contacts
BC7: Referrals	Introductions where one or more party must be endorsed, authorised, and or made aware of by another.
BC8: Reputation	The reputation is a global score for participants representing trustworthiness.
BC9: Bookkeeping	Recording of transactions, typically for the purposes of financial reporting.
BC10: Brokering	Introductions for asset-transfer contracts.
BC11: Monitoring	The automated detection of transactions or contract executions satisfying particular, pre-defined properties.
BC12: Offering (incl. auctions)	Contract / transaction with initially undetermined counterparty.

may be interrelated (for example, settlement will involve payment). In addition, the business capabilities we have focussed on are multi-party capabilities, rather than capabilities that are mainly internal to a company.

#### 4.2.4 Features of Blockchain-Based Systems

We then identified a list of blockchain features (system capabilities) as outlined in Table 2.

We then mapped the different business capabilities to the corresponding blockchain features (cf. Table 3). Optional features are listed in brackets.

A combination of the above described capabilities can be used to form a market.

Additionally, we identified the following inter-dependencies among features.

1. Audit Trail → Transactions → Signature → Encryption → Wallet information;
2. Contract → states → Verification;

■ **Table 2** Features of blockchain-based systems.

F1: Data access on-chain	Storage; universal access to data stored on the ledger for any processing node.
F2: Encryption	Ability to encrypt and decrypt data stored on the blockchain.
F3a: Channel	Need-to-know access to data. Access control list.
F3b. Vault/Wallet information	Access to private information necessary to operate on the blockchain, but should remain confidential (e.g. private keys).
F4: States	Ability to record state for assets defined on the ledger, and transition the states using smart contract executions.
F5: Audit trail	Ability to record and link events in a sequence (provenance, logging, states are chained).
F5b: Receipts	Ability to obtain a detailed record per transaction, indicating which assets were read and modified.
F6: Transactions	Ability to submit transactions.
F6b. Permissions to submit data on-chain. F7. Identity management	
F8: Contract	Ability to invoke programs through transactions, and store contracts on-chain.
F9: Process	
F10. Verification	Integrity check of the ledger, and contract execution.
F11: Time service	Authoritative source of physical time, and timestamping.
F12: Notary service	Ability to put trust in / responsibility for a particular computation step in a given participant.
F13: Oracles	Special case of notary which injects external information into the system. (A mechanism for ensuring integrity of data provided transparently by a trusted data source.)
F14: Tokens	
F15: Anonymization	
F15b: Pseudonymization	
F16: Watermarking	Ability to permanently fix a signature inside a document stored on the chain.
F17: Digital signature	Ability to attach a signature to a transaction / document on the chain.
F18: Event	Ability to send events between accounts, to trigger smart contract invocations, and to notify external subscribers.
F19: What-if analysis	Ability to query the projected impact of a transaction / contract execution on the current state of the blockchain [9].

■ **Table 3** Mapping Capabilities to Features.

Capabilities	Features
Voting	Transaction, Time service, (Anonymization, Notary, Identity Management, Tokens)
Payment	Transactions, Receipts, (Channel, Time service, Tokens)
Asset transfer	Transactions, Tokens, Watermarking, (Channel)
Settlement	Audit trail, Tokens, Notary, Contract
Exchanges	Transactions, Tokens, Assets transfer, Notary
Introductions	Process, Data access, Channel
Referrals	Transactions, Tokens, Identity Management
Reputation	Identity Management, Audit Trails, (Oracles)
Bookkeeping	Audit trails, Receipts, States
Brokering	Identity, Contract, Transactions, State, What-if
Monitoring	Audit trail, Events, Process, Contract (Time Service)
Offering (incl. auctions)	Transaction, Contract, Digital signature, (Time service, pseudonymization)

3. Time service → oracle → Notary;
4. Channel → Identity management → Encryption;
5. Tokens → Transactions.

#### 4.2.5 Conclusion

This summary has taken initial steps towards identifying both the features that can reasonably expect to be supplied by a blockchain platform on the one hand; and the capabilities which applications for that platform may require on the other.

In the future we would like to investigate which features are supported by different blockchain platforms, to guide the decision which platform to choose.

Moreover, as another avenue of research we might look into different solution patterns on how to implement different features.

### 4.3 Factors Influencing Process Analytics on Distributed Ledgers

*Claudio Di Ciccio (Wirtschaftsuniversität Wien, AT), Luciano García-Bañuelos (University of Tartu, EE), Mieke Jans (Hasselt University, BE), Jan Mendling (Wirtschaftsuniversität Wien, AT), Petr Novotny (IBM TJ Watson Research Center – Yorktown Heights, US), Ludwig Stage (Tübingen, DE)*

**License** © Creative Commons BY 3.0 Unported license  
 © Claudio Di Ciccio, Luciano García-Bañuelos, Mieke Jans, Jan Mendling, Petr Novotny, Ludwig Stage

Blockchains trace the sequence of tasks carried out in the course of business process executions by the totally ordered recording of transactions between involved parties, and additionally the logs of events registered by Smart Contracts. This leaves ample room for the ex-post analysis of conducted operations, for analytics, auditing, and mining purposes [40]. However,

it also poses questions on how to design the data storage, keeping into account a.o. the following facts: firstly, the recording of information, or the absence thereof from the stored data, influences the knowledge that can be extracted from ledgers; secondly, the exchange of data in blockchains such as Ethereum is expensive both for what the transactions fees and the write operations from Smart Contracts are concerned [66]; thirdly, saving information as data values carried as transactions payload entails a better traceability, on the one hand, but also unlimited access to the possibly sensible information exchanged, which is detrimental from a privacy viewpoint, on the other hand; finally, multiple blockchains can be adopted that can be either homogeneous or heterogeneous, e.g., a federated network of Ethereum ledgers (e.g. Quorum<sup>1</sup>) for the general inter-organisational, multi-party collaboration, and a federated Hyperledger Fabric for sub-processes involving sub-groups among the participants. On top of this, the introduction of querying languages for data stored as transactional or logging information plays a pivotal role in the introduction of process analytics over blockchains. To that extent, the preliminary contributions provided by the state query languages of Hyperledger Iroha <sup>2</sup>, Hyperledger Burrow <sup>3</sup>, and the querying of the backing MongoDB database through EOS <sup>4</sup>, provide promising results.

In this report, we focus on challenges and requirements for conducting business process analytics on data stored by blockchain-backed process management systems. In particular, we examine the cases in which information is stored fully on-chain or partially off-chain.

### 4.3.1 On-chain challenges and requirements

In this section we investigate the case in which all the information that is relevant to the process execution is stored on-chain. Discussions on data management and provenance associated with this strategy, including the privacy concerns and the transaction costs, go beyond the scope of this summary. Even under the assumption that the issues related to those topics were appropriately handled, we envisage in the following some crucial aspects to be taken into account for the analysis of this data.

#### Audit-completeness

Starting with the fully on-chain, single ledger design, the audit-completeness of this data is paramount. Taking inspiration from a requirement set by process mining, if criteria are not provided to uniquely identify the transactions pertaining to a process instance, then linking the evolution of the process becomes a hard manual work at best, thus hampering the process analytics endeavours [6]. In the context of financial auditing, the auditor needs to consider both relevance and reliability of audit evidence. In the context of using blockchain technology, both dimensions might be impacted. The data that is stored on the blockchain ideally contributes to financial reporting assurance (relevant data). Further, providing evidence that data is sufficiently reliable, is often challenging to the auditor [46]. When evaluating this aspect, accuracy and completeness of the data are considered; two aspects that may be impacted positively impacted by blockchain.

---

<sup>1</sup> <https://www.jpmorgan.com/quorum>

<sup>2</sup> <https://www.hyperledger.org/projects/iroha>

<sup>3</sup> <https://www.hyperledger.org/category/hyperledger-burrow>

<sup>4</sup> <https://eos.io/>

### Eventual consistency

As explained in the CAP theorem [10], distributed systems can enjoy at most two properties out of Consistency (every read receives the most recent write or an error), Availability (every request receives a non-error response), and Partition tolerance (the system continues to operate despite an arbitrary number of messages being dropped or delayed by the network). Reportedly, for instance, Ethereum does not guarantee (strong models of) consistency [4], but only eventual consistency [64]. This signifies that the monitoring of transactions carried out on a local node does not guarantee full reliability. We identify in this context the following *audit patterns of deviation*:

**Reordering** Transactions, as well as the data they bring, could occur re-ordered in case the local world state in a node gets changed by the substitution of the latest block, or a sub-chain, with a fork that achieved a larger consensus;

**Recurring** Supposing that a fork lead to a local history rewriting, an already analysed block could be replaced with a new one in which a processed transaction does not occur any longer, yet it recurs in a new mined block thereafter; in such a case, the same information might be included twice if a consistency check is not operated that rearranges the parsed information accordingly;

**Missing** In the case of forks, transactions that were considered as valid could be excluded by the agreed-upon fork, and then not re-included in case new transactions mined in the new blocks make them invalid; this poses the challenge on whether to discard the retrieved information when the corresponding transaction gets erased from the blockchain.

### Abstraction and reverse engineering

In Ethereum, information stored on-chain can occur as event logs emitted from Smart Contracts or as payload to transactions, aside of the internal state of contracts which is however not explicitly written on transaction receipts but has to be recomputed by executing the code locally to the nodes in order to be undisclosed. Event logs and data parameters of the transaction can reveal explicit notifications and context specifications respectively, upon deserialisation<sup>5</sup>. Nevertheless, the way in which logs and exchanged data are engineered is tightly bound to how the the Smart Contracts are encoded. This hampers the ex-post interpretation of those sources of information, let alone their automated analysis. The promised verification and traceability of executed processes ends up being ad-hoc, and demanding manual effort, not so differently from what used to happen when striving to understand the behaviour of legacy systems through their logs [47]. This calls for the introduction of a specification language, possibly using the decorator pattern [22], to enrich the code of methods in Smart Contracts for the sake of self-documentation about state, data, and logging.

#### 4.3.2 Off-chain challenges and requirements

In the remainder, we examine the case in which data are held out of the ledger. Again aside of the considerations on the data management and administration, we portray the envisioned challenges and requirements that involve the merging of operational information retrieved from the blockchain and the affected data from the outside.

---

<sup>5</sup> <https://solidity.readthedocs.io/en/develop/abi-spec.html>

### Between ledger and the outer data

As the information is split between on-chain and off-chain data, the analysis necessitates of a mechanism to join at least two sources of information, one logging the sequence of actions mediated by methods invoked on smart contracts, the other reporting on the updates on, or retrieval of, data witnessing the conducted tasks. Technologies such as the InterPlanetary File System (IPFS) provide a mechanism for uniquely linking data chunks spread among peers outside the main blockchain. However, here we refer to those cases in which parts of the data pertaining to the process activities are kept in other systems that can be disconnected from the ledgers, such as external DBMSs, either centralised or federated. Especially in such a case, the association of ontologies to the process specification seems paramount to describe the semantic connections.

### Versions of blockchain artifacts

Should the process undergo a redesign phase, the Smart Contracts implementing the old version could be replaced by newer ones. However, whilst the versioning of processes is a feature that is implemented by current Business Process Management Systems (BPMSs), the concept of contract replacement is not natively supported by blockchains such as Ethereum. Fully on-chain software architectures such as the one of the blockchain-based BPMS Caterpillar [37] may cater for it, thanks to their implementation of the factory pattern for generating Smart Contracts. However, in partial on-/off-chain scenarios, keeping track of the changes entailed by subsequent versions of the involved artifacts becomes a challenge of higher difficulty, as it involves at once information integration and object matching, together with the semantic version control over software and data updates.

### Between ledger and reality

Solutions to connect the digital world of the blockchain with outer reality are crucial to cater for business processes interacting with physical objects, such as in the case of the manufacturing, logistics, or healthcare domains, to mention but a few. For instance, the notion of time is implemented on blockchains such as Ethereum as *block-time*. Such a timestamp is shared among all transactions therein, thus at a coarse-grain level. The aforementioned business processes operate in real time instead. To solve this problem and inject information on real-world information including time, so-called *in-bound oracles* such as Oraclize<sup>6</sup> have been introduced. Oracles operate as a middleware connecting reality with the on-chain information space, and take on the task to return consistent answers to virtually all nodes in the blockchain that execute the same Smart Contract locally. The out-bound connection seems however more challenging. Owing to the concept of eventual consensus, an operation executed by the run of a Smart Contract may be withdrawn, should a different suffix of the blockchain eventually reach consensus over the local version. From a BPM perspective though, the execution of certain tasks should not be subject to rollback, especially if it leads to permanent changes in the real world. Waiting times could be introduced on purpose between the local transaction and the actual execution of the associated operation in real world, so as to reduce the probability that the task is undone. This is indeed what happens with many purchases paid in Bitcoin. However, this approach might lead to delays that encumber or disrupt the executions of business process in general. Besides, only mitigating the uncertainty of task executions is not sufficient in many cases. An approach that eliminates the risk of operation rollback is thus of high relevance and calls for future research endeavours.

---

<sup>6</sup> <https://github.com/oraclize>

#### 4.4 A Holistic Vision of Blockchain-Based Application Design, Specification, and Implementation

*Michael Coblenz (Carnegie Mellon University – Pittsburgh, US), Richard Hull (IBM TJ Watson Research Center – Yorktown Heights, US), Qinghua Lu (Data61, CSIRO – Sydney, AU), Ingo Weber (Data61, CSIRO – Sydney, AU)*

License © Creative Commons BY 3.0 Unported license  
© Michael Coblenz, Richard Hull, Qinghua Lu, Ingo Weber

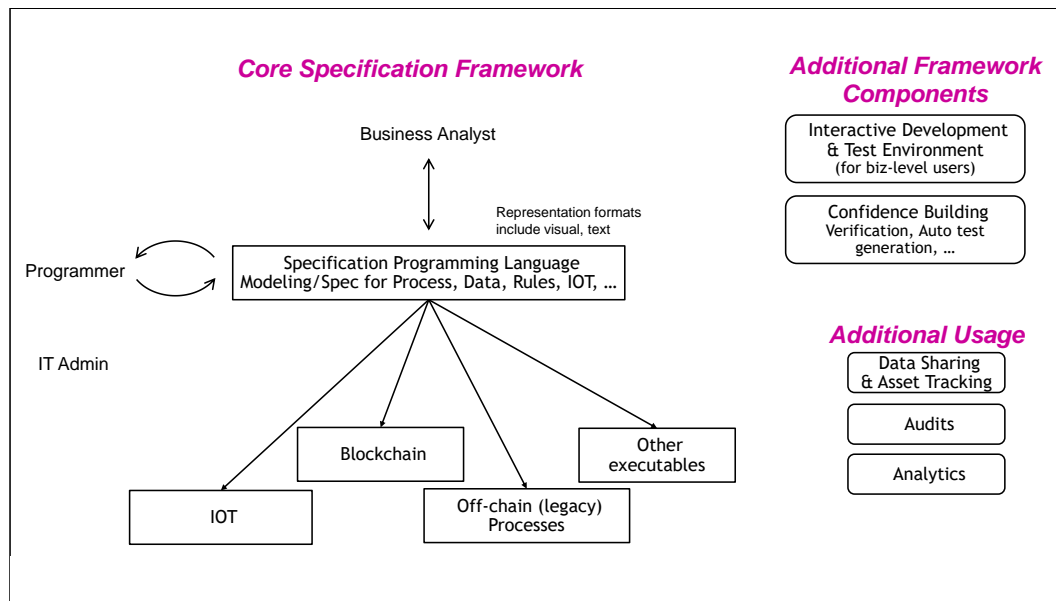
Blockchain is an emerging technology, and we are just at the starting point in understanding its potential and where it can bring real value. The existing design and specification methods are suboptimal for blockchain-based applications, and not well integrated – e.g., BPMN does not support data well enough for many use cases on blockchain; on-chain and off-chain parts of the application are viewed disparately (and this separation is unlikely to resolve, due to the nature of some data and computation being required to remain off-chain); and governance for the evolution of blockchain smart contracts brings challenges not found in other contexts. Our vision is to have a holistic design/specification/implementation approach which covers various disparate components in an integrated fashion.

##### 4.4.1 Problem Statement

Background: Blockchain is a new distributed ledger technology, which has attracted broad interest, including in industry and government, in exploring how to use blockchain to re-design systems and to build the next generation of applications. Issues include the following:

1. The existing design and specification approaches are suboptimal and not well integrated. First, business analysts usually lose control of system development once they complete requirement specification since they can hardly understand the smart contract programming language and check the conformance with the requirements they specify. It is also hard for domain experts to inspect the code to understand how their ideas are represented in the system. Second, designers need to work on different types of model abstractions which are not well integrated. Third, blockchain is by design data-centric, which can hardly be supported by existing models, e.g. BPMN does not support data modeling well.
2. The architecture of blockchain-based system is usually separated into on-chain and off-chain parts since blockchain has limited storage capability and the information on blockchain is designed to be accessible to all the participants. This separation of on-chain and off-chain is unlikely to be resolved completely, due to the nature of some sensitive and large sized data, compliance requirements, and some computation being required to remain off-chain (in part due to the high cost of on-chain computation).
3. Blockchain systems raise challenges in governance not typical in other domains. These systems will increasingly be designed and maintained through collaborations of multiple stakeholders coming from different organizations. Agreed upon smart contracts are themselves stored on the Blockchain to support increased trust in the system. Furthermore, because the data is immutable, there is an increased desire for reverse compatibility of smart contract versions. This to enable easier usage of data from earlier versions of a smart contract, and in auditing and analytics solutions layered on top of the Blockchain.





■ **Figure 2** High-level framework for design, specification, and implementation of Blockchain-enabled solutions.

#### 4.4.2 Proposed Approach

Our vision is to have a holistic approach for the design, specification, and implementation of collaborative information systems using blockchain, which covers various disparate components in an integrated fashion. For instance, there might be a set of models for static and dynamic aspects (e.g., data, process, business rules, required inputs from user and devices) which is translated into executables for blockchain (smart contracts), enterprise systems (including DBMS schemata and internal processes / rules), IoT device instructions, and UIs. By starting from the beginning, we can leverage the characteristics of the blockchain. We propose a technique that uses research methods in human-centered design in order to maximize the chances that the resulting system will be as effective as possible for its users. By doing so, we will support different roles, including business analysts, domain experts, and programmers, with one holistic method. Reasons for this requirement include the need of business users to own and understand the application well enough so that they can ensure that the resulting implementation conforms to their expectations, and they can assess risks, compliance, and likely or probable effects.

#### 4.4.3 Research Questions

##### User-oriented questions

A solution should address the following general categories of users, representing several different collaborating organizations:

- Business analysts
- Programmer/software engineers
- IT administrators
- End users
- Lawyers

In order to assess the needs of the users, we should address the following questions:

- What use cases do systems need to support for each user?
- How do users expect to express solutions to their problems in their own domains?
- In order to make tangible progress in the shorter term, what modeling paradigms (e.g. BPMN, CMMN, etc.) can be adapted and integrated to form a first workable programming model/language?
- What principles can guide creation of usable integrated development environments for business analysts?
- How can we characterize the domain of applications that need to be supported? Is existing work in this area sufficient? Academic work on blockchain may not capture enough of the business use cases, and industrial efforts to date may not be publicly accessible or represent the whole range of possible use cases for the tool set.

### Multiple target platforms

An individual blockchain application may comprise components for more than one platform, such as:

- Blockchain programs
- IoT
- Off-blockchain server-side programs
- User-facing systems (e.g. the UI for a dapp)

This leads to the following research questions:

- How can a system facilitate interoperation among the different components? For example, data structures may be passed among different components; we want to ensure consistent semantics for a given object/asset.
- To what extent can portions of the application be targeted at executables in a flexible way? For example, an initial prototype might run entirely off-chain, with components moving to the blockchain as needed or as scalability is demonstrated. Or an application might assume that many IoT devices are going to provide data, and later the data may be sourced via traditional transaction invocations.
- Do the different targets need distinct kinds of specifications/implementations? Is it appropriate to consider UI as part of the blockchain application, or is it more properly considered as a separate project that invokes APIs?

### Underlying blockchain platforms

What are the relevant properties of blockchain platforms that vary among platforms (e.g. Hyperledger Fabric, Ethereum, etc.)? How are applications customized/optimized to run on particular platforms? For example, how should an application designer/implementer decide how to represent the state need to be serialized to the ledger?

How do governance needs affect application development needs? For instance, how can blockchain applications evolve over time?

### Modeling-oriented questions

The model-driven community tends to assume that it is best to separate a model of the system from the implementation, which may be partly or completely generated from the model. However, this separation might unnecessarily add complexity to the system (requiring some or all users to understand two different perspectives of the system and keep them in

sync). Further, it might result in mismatch between people's expectations (in the model) and the reality of the lower-level implementation.

- To what extent should the users work on the implementation directly versus work on several different abstractions of the system?
- What kinds of views facilitate the reasoning that different stakeholders need? Which of these views need to be editable, and in what cases might higher-level changes accidentally invalidate previous lower-level modifications?
- To what extent can we re-use existing modeling paradigms (e.g., BPMN, CMNN, UML class diagrams, SBVR, etc.) vs. develop variants and or new paradigms? If some existing modeling paradigms are used, how can they be combined together (e.g., how to combine the process-centric perspective of BPMN with the data-centric perspective of UML class diagrams?)

### Analysis/auditing

Blockchain provides the opportunity to hold a shared global view of all data relevant to a collaboration between organizations. This data can be used in ways that enterprise-specific data has been used in the past, including to track assets, support audits, and perform analytics. This raises several research questions, including the following.

- What use cases should auditing tools support? What are the needs of the users, and who are they?
- What design principles should be followed with regards to data and processing, in order to enable conceptually simple asset tracking, auditing, and analytics? This should permit efficient implementation on various blockchain technologies, with regards to execution of smart contracts and of the data processing for tracking, auditing and analytics.
- Some blockchain technologies include privacy and access controls concerning both data and processing steps. How do these interact with support for tracking, auditing, and analytics?

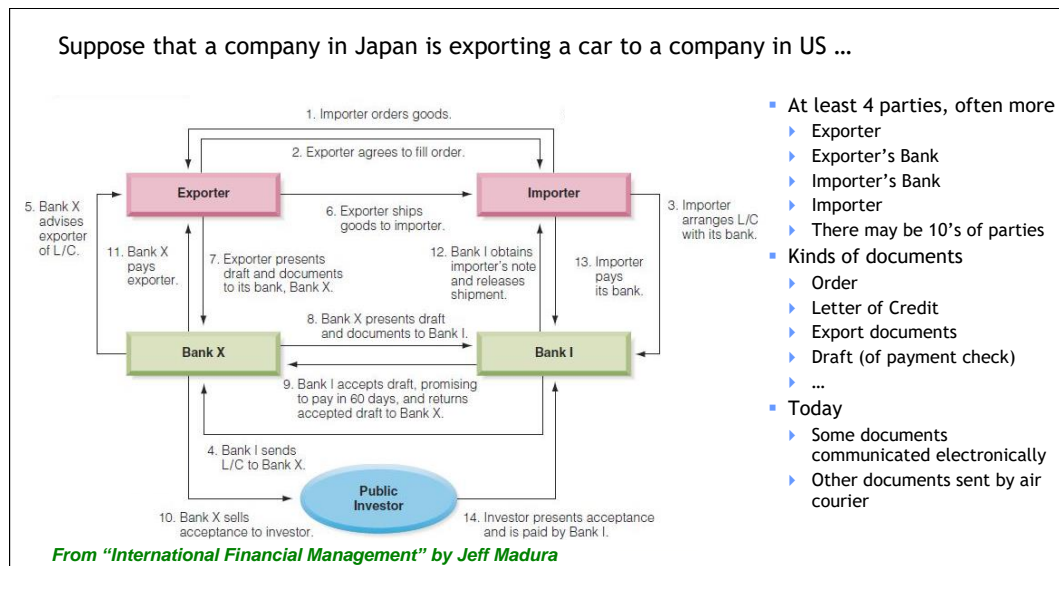
### Evaluation

We did not design a particular evaluation plan. An evaluation plan would need to address these questions:

- What kind of user studies would provide evidence regarding usability from the various perspectives?
- What case studies should be done to evaluate expressiveness and to help iterate on the system design?
- Are there formal properties of the system that should be verified?

#### 4.4.4 Summary

This working group discussed a vision of a holistic approach for design, specification, and implementation of blockchain-based information systems. The main part of the discussion focused on research questions, covering user-oriented aspects, target platforms across blockchain and off-chain components, IoT, and UIs, blockchain platforms, modeling, analysis, and evaluation aspects. Achieving this vision would require major effort, but could yield many benefits over existing solutions.



■ **Figure 3** Simplified view of Trade Logistics use case.

## 4.5 Towards a Blockchain Collaboration Meta-Model and Language

Michael Coblenz (Carnegie Mellon University – Pittsburgh, US), Claudio Di Ciccio (Wirtschaftsuniversität Wien, AT), Marlon Dumas (University of Tartu, EE), Fabiana Fournier, Luciano García-Bañuelos (University of Tartu, EE), Richard Hull (IBM TJ Watson Research Center – Yorktown Heights, US), Qinghua Lu (Data61, CSIRO – Sydney, AU), Raimundas Matulevičius (University of Tartu, EE), Jérôme Siméon (Clause Inc. – New York, US), Mark Staples (Data61, CSIRO – Eveleigh, AU), Ingo Weber (Data61, CSIRO – Sydney, AU)

License © Creative Commons BY 3.0 Unported license

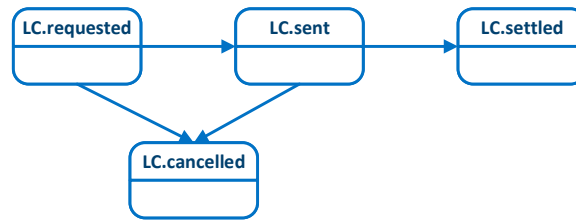
© Michael Coblenz, Claudio Di Ciccio, Marlon Dumas, Fabiana Fournier, Luciano García-Bañuelos, Richard Hull, Qinghua Lu, Raimundas Matulevičius, Jérôme Siméon, Mark Staples, Ingo Weber

A blockchain collaboration is a process involving multiple *participants* that interact between them by means of a blockchain. An execution of a collaboration (a *collaboration instance*) requires binding roles in the collaboration to particular participants. The participants create and manipulate a number of *concepts*. A *concept* is a container of data (possibly stored on the blockchain), such as Purchase Order, Invoice, Product, Letter of Credit, etc.

A concept is either an *asset* when it is subject to a (linear) ownership relation with the possibility of ownership transfer, or a *business object* otherwise. An *asset* has a number of properties, which may be attributes (e.g. the amount of an invoice), asset-to-asset relations, or asset-to-participant relations. An example of an asset is a house, a car, a parking place, or a product. A purchase order or an invoice are examples of business objects. Note that it is possible that a business object (not subject to ownership) can have a corresponding tradeable invoice, which is an asset and may thus be transferred. In this case, we distinguish the tradeable invoice as an asset and its corresponding invoice (business object) from which it originates.

Concepts may be fungible if they do not have an identity, or non-fungible in which case each instance of the concept has a unique identifier.

A business collaboration consists of a set of concepts, a set of participants, and a set



■ **Figure 4** Letter of Credit lifecycle.

of transactions. A transaction modifies the state of the collaboration, which may imply modifying the properties/state of one or more concepts in the collaboration.

The occurrence of transactions may be constrained by the *collaboration's behavior*. At an abstract level, a collaboration behavior determines/restricts whether or not a given transaction can take place given the current state of the concepts involved in a collaboration.

The behavior of a collaboration may be defined at a local level (i.e. at the level of one asset at a time) or at a global level (i.e. at the level of a collaboration consisting of multiple assets). Below, we sketch multiple approaches for modeling collaboration behaviors both at a local and at a global level. To illustrate each of these approaches, we make use of the trade logistics scenario depicted in Figure 3.

Three of the approaches are tied to a model-driven paradigm and include both visual and machine-readable specification (Sections 4.5.1, 4.5.2, 4.5.3). Two are focused on incorporating Blockchain-relevant constructs and abstractions into domain-specific programming languages (Sections 4.5.4), 4.5.5). There are other somewhat related approaches for orchestrating or choreographing the assets that are not considered here, e.g., the declarative style of object-centric behavioral constraints (OCBC) [61], the declarative choreographies for business artifacts [56], and the more procedural BPMN process choreographies [20]. Section 4.5.6 builds on some of these ideas to develop a framework for specifying collaborative logs in the context of Blockchain solutions.

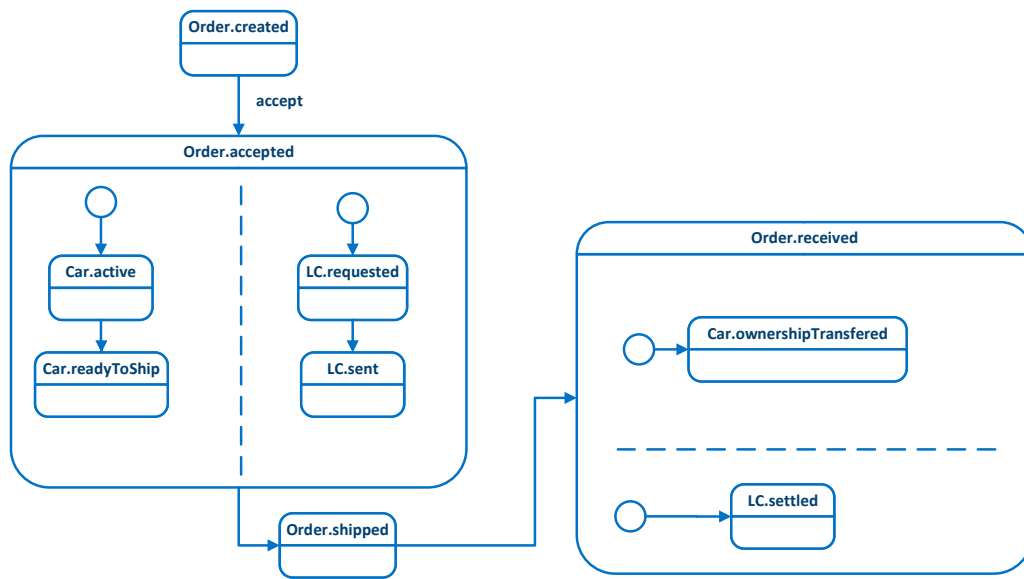
#### 4.5.1 A Statechart-Based Approach

In this approach, the behavior is modeled both at a local and at a global level using statecharts.

Each asset has a set of possible states, which may be related by means of a statechart diagram. For instance, a letter of credit has four states: “requested”, “sent”, “settled”, and “cancelled” – with the lifecycle shown in Figure 4. Assets can also have different types of status (with a lifecycle each), and constraints can exist between them. For instance, a car can be in high-level states “active” and “inactive”, and its loan status can be “owned outright” or “collateral to loan”. In addition, a car may be in state “ready to ship”. Transitions are labeled, e.g., a self-link from and to “owned outright” called “transfer ownership”.

Because we are on a blockchain, there is no separation between Purchase Order and Sales Order; instead, the shared view between the parties is held on-chain, and has two associated roles: buyer and seller.

In addition to the local behavior (local statechart), we have one global state chart for the whole blockchain collaboration, which describes how the different lifecycles are connected to achieve the goal of the business collaboration. For instance, consider the global state chart in Figure 5, which proceeds as follows. The collaboration starts with an order being



■ **Figure 5** Global statechart.

created by the buyer, and subsequently accepted by the seller. Then, in parallel, the car is manufactured (“car.created”) and in parallel the Letter of Credit (LC) is requested from the importer’s bank (“LC.requested”).

#### 4.5.2 A BPMN-based Approach

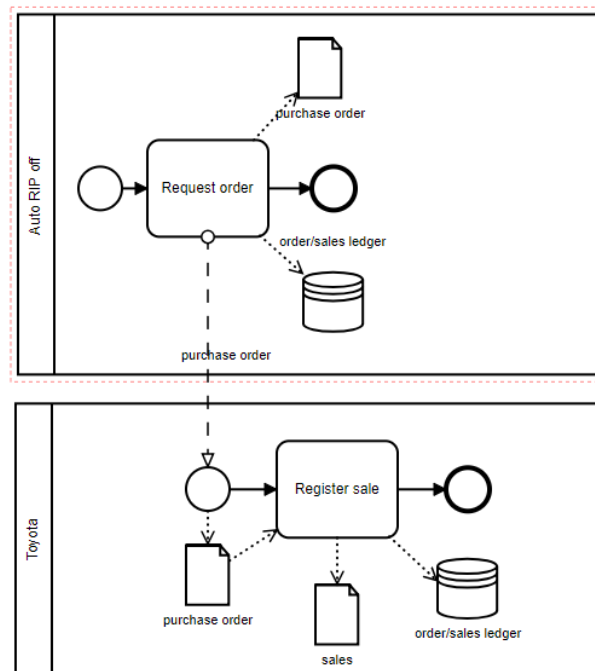
In this approach, the behavior of the collaborative process is captured as a business process model in the standard BPMN notation. In this model, lanes represent Parties which collaborate together via a blockchain platform. The transactions that parties can execute on the blockchain are captured as activities in the BPMN process model. The behavior is thus captured via sequence flows, gateways, and other BPMN control-flow constructs.

The running example, captured using this approach, is shown in Figure 6 (specifically this model captures the collaboration between an exporter e.g. Toyota and an importer e.g. Auto Rip Off. The data objects manipulated in the process includes the Purchase Order (from Auto Rip Off’s point of view) and the Sales Order (from Toyota’s point of view). Transactions (Request order and Register sale) are written to ledger, which represented as a data store Order/Sales ledger in the model.

BPMN-based approaches have been proposed in the literature already, including [23, 65], and implemented in the tools Caterpillar [36] and Lorikeet [58]. Lorikeet has asset management and control features, which can be linked to process activities.

#### 4.5.3 An approach inspired from Case Management

In this approach to the global orchestration across assets, parties, and their relationships, we focus on a modular model of *activities* along with a flexible, somewhat declarative approach for specifying when activities might be launched and concluded. In one sentence, the global structure of the activities follows the structure of a (hierarchical) Directed Acyclic Graph (DAG) where the control of activity launch and completion is controlled by a combination of events and conditions that refer to the global state. This follows the spirit of case modeling



■ **Figure 6** A BPMN-based process model.

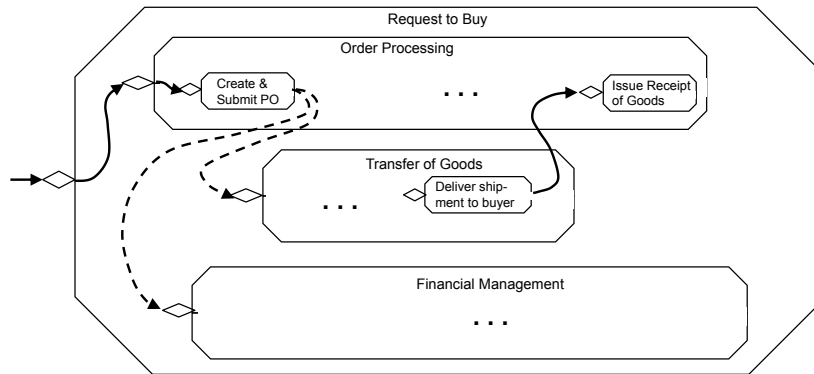
in general (cf. OMG CMMN), the van der Aalst et al. Case Handling paper [62], and work on the Guard-Stage-Milestone (GSM) artifact-centric model [28, 18]. A formal operational semantics for this model can be developed along the lines of GSM, but would be much simpler because the focus is on DAG with rollback, rather than full GSM. A more detailed description and illustration now follows.

Figure 7 provides a high-level, simplified view of how the DAG with rollback approach might be visualized for the trade logistics example. The focus of this view is on the activities; the linkage to assets and their lifecycles, and to the relationships between assets and parties is not incorporated here. The notation follows that of CMMN, but with some variations.

The key building blocks are activities, including *atomic activity*, *long-running activity*, and *composite activity*. Activities are launched by events. The events might come from the outside world (i.e., transaction requests into the Blockchain), or may be *internal*, by which we mean triggered from the completion of some other activity in the orchestration. The activities may include *guards*, which are conditions on the state of the underlying assets and relationships, and also the state of the global orchestration. The solid arrows indicate events, including some internal ones, and the diamonds indicate guards. The diagram illustrates that the activities can be nested.

Rollbacks may occur, following the spirit of van der Aalst's Case Handling model, but extended to include modularity (and possibly some notion of compensating actions).

A key aspect of the DAG with rollback approach is the rich flexibility that can be incorporated with regards to the ways that activities might occur in sequence and in parallel.



■ **Figure 7** Illustration of DAG with rollback for global orchestration.

#### 4.5.4 A legal contract perspective

Legal contracts are collaborative processes involving multiple participants, usually in a context where there is a measure but not entire trust between the parties. As such they are an interesting model, and an important use case, for Blockchain applications.

A legal contract typically comes with a natural language describing the terms of the agreement, its conditions and the process that is allowed or mandated between the parties. However, one can also consider a legal contract from an information system point of view. In this *legal contract perspective*, the contract is the central object being created and executed. This usually happens in three steps.

The first step is the negotiation phase during which the parties discuss the terms of the contracts, until they reach an agreement on those terms. From an information system point of view, this is similar to agreeing on the logic (or program) that will be executed.

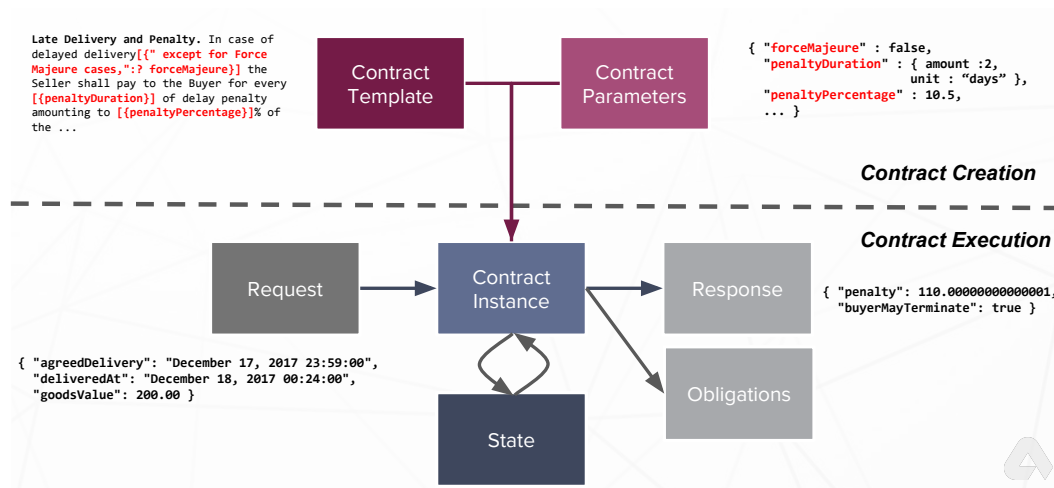
The second step is to sign and *execute* the contract. From an information system point of view, this is similar to creating an instance of the logic (or program) that has been agreed upon.

The third step is contract execution, where the contract serves as an agent between the parties. During that step, the contract typically maintains a state and respond to requests (e.g., what is the price that should be paid for the goods delivered, including penalties if any).

Figure 8 gives an overview of this approach. In the top half of the figure is the contract instantiation, which involves creating a contract (e.g., from a contract template which has been negotiated and adapted for the specific parties and business needs). Once the contract has been instantiated, it can be executed. The execution involves sending requests to the contract, which returns a response and may change its state.

In this section, we present a purchase agreement between a car dealer and a car manufacturer based on the legal contract perspective. This example is loosely inspired by the one written in Obsidian that we saw in Section 4.5.5.





■ **Figure 8** A Legal Contract Perspective.

### Modeling the data

To model the data manipulated in a blockchain-based collaborative process, one can use the Composer Modeling Language (or CML) offered by the Hyperledger Composer platform.

This is a convenient way to describe a class hierarchy with some specific distinctions relevant to contracts (e.g., it distinguishes between general purpose concepts, participants which are the parties involved in the contract, and assets which can be owned and transferred between parties).

```
namespace org.dagstuhl.automotive

import org.accordproject.cicero.runtime.*

// Car and inventory
asset Car {
  o String name
  o Double purchasePrice
}

concept Inventory {
  o Car car
}

// Participants
participant Dealer {
  o String name
}

participant Shipper {
  o String name
}

participant Manufacturer{
  o String name
}
```

```

o Shipper shipper
o Inventory inventory
}

// Contract parameters
concept Parameters{
o Dealer buyer
o Manufacturer supplier
}

// Contract state
asset SupplyAgreementState extends AccordContractState {
o Car car
o Participant owner
}

// Contract transactions (request/response)
transaction Order extends Request{
o Car car
}
transaction Shipped extends Response{
o DateTime at
}

transaction Deliver extends Request{
}
transaction PaymentDue extends Response{
o Double amount
}

transaction Pay extends Request{
o Double amount
}
transaction Completed extends Response{
o String message
}

```

Note that the model includes parameters of the contract (i.e., which are the parties in this specific example), and the structure of the contract state (here which car is being purchased and which current participant owns the car).

This is only a simple example, one could model something much more complex, for instance the various parties here could be US Businesses with certain general information (where is it incorporated, etc). But for now this will do.

### Describing the logic

The next step is to describe the logic of the contract. Here this is a very simple contract the contract is between a buyer (of type `Dealer`) and a supplier (of type `Manufacturer`).

There are only a few simple operations available in this contract:

1. The buyer can place an order and the contract responds with shipping information from the manufacturer
2. The shipper can deliver the car and the contract response with payment information

- The buyer can pay the bill and the contract responds with a thank you note (or screams if the payment is not correct)

The following code for the contract is written in Ergo:

```
contract SupplyAgreement over Parameters state SupplyAgreementState{
// Initialize the contract
clause init(request:Request) {
set state SupplyAgreementState{
stateId: "IDLE",
car: contract.supplier.inventory.car,
owner: contract.supplier
};
return
}
// Place an order
clause order(request:Order) : Shipped {
enforce(request.car.name = contract.supplier.inventory.car.name);
set state SupplyAgreementState{
stateId: "INTRANSIT",
car: state.car,
owner: contract.supplier.shipper
};
return Shipped{ at: now() }
}
// Delivery clause
clause deliver(request:Deliver) : PaymentDue {
set state SupplyAgreementState{
stateId: "DELIVERED",
car: state.car,
owner: contract.supplier
};
return PaymentDue{ amount: state.car.purchasePrice }
}
// Payment clause
clause pay(request:Pay) : Completed {
enforce request.amount = state.car.purchasePrice;
set state SupplyAgreementState{
stateId: "COMPLETED",
car: state.car,
owner: contract.buyer
};
return Completed{
message: "Enjoy your " ++ state.car.name
}
}
}
```

As the reader can observe, there is a contract structure (akin to a class in an Object-Oriented programming sense), and a contract contains clauses (akin to a method in an Object-Oriented programming sense). There is a special `init` clause to set up the contract (akin to a constructor method in an Object-Oriented programming sense). In our example contract, we have one clause per operation available on the contract (one to order, one to deliver and one to pay).

In Ergo, the `return` statement indicates the response when calling a clause, the `set` statement indicates a change of state in the contract, and the `enforce` statement is a precondition for the clause (i.e., the clause with return to the caller with an error if it is false).

### Instantiating and invoking the clauses of the contract

Now that we have modeled our data and our contract, we are ready to instantiate the contract. This can be done with the following Ergo declarations:

```
// Let's create all the parties
define constant the_car = Car{
  name : "Athena 360",
  purchasePrice : 36000.00
}

define constant the_manufacturer = Manufacturer{
  name : "AutomotiveInc",
  shipper: Shipper{ name : "HappyShipping" },
  inventory : Inventory{ car: the_car }
}

define constant the_dealer = Dealer{ name : "Best Deal Bros." }

// Now we can initialize one contract
set contract SupplyAgreement over Parameters{
  buyer : the_dealer,
  supplier : the_manufacturer
}
call init(Request{});
```

Now we are ready to ship cars! Here is an example of calling the contract from purchase order to completion. At each steps we show the response and the new state of the contract along with their types.

```
call order(Order{ car: the_car });
Response. Shipped{ at: dateTime("2018-08-16 16:16:42")} : Shipped
State. SupplyAgreementState{
  stateId: "INTRANSIT",
  car: Car{name: "Athena 360", purchasePrice: 36000.0},
  owner: Shipper{name: "HappyShipping"}
} : SupplyAgreementState

call deliver(Deliver{});
Response. PaymentDue{amount: 36000.0} : PaymentDue
State. SupplyAgreementState{
  stateId: "DELIVERED",
  car: Car{name: "Athena 360", purchasePrice: 36000.0},
  owner: Manufacturer{...}
} : SupplyAgreementState

call pay(Pay{ amount : 36000.00});
Response. Completed{message: "Enjoy your Athena 360!"} : Completed
State. SupplyAgreementState{
  stateId: "COMPLETED",
  car: Car{name: "Athena 360", purchasePrice: 36000.0},
```

```
owner: Dealer{name: "Best Deal Bros."}
} : SupplyAgreementState
```

#### 4.5.5 Languages for direct editing

Some of the approaches in this document separate a meta-model from the programming language. In this section, we describe a different vision: choose one language but potentially provide visual editors. Such an editor might leverage some users' familiarity with prior meta-modeling tools, but with an important difference: the editor would edit the program directly rather than editing a model of the program, which might be semantically disconnected from the program. It might also facilitate code understanding and architectural analysis by providing diagrams that enable understanding of high-level behavior (rather than requiring readers to infer relationships between components).

By having a direct visual editor for the language, the program is always consistent with the visual representation, and there is the opportunity for gradual learning of the language. In meta-modeling systems, the user may be required to learn two different languages and edit them separately.

#### RequestToBuy: Roles, assets, and relationships example

This approach is intended to be directly representative of the approach we discussed, in terms of distinguishing roles, assets, and participants, mediated by transactions.

A role describes an entity that can have behavior, whereas assets are merely containers for data. Transactions can be invoked externally; in contrast, an *action* can be thought of as a step in a process. These declarations enable tools to infer high-level process structures (perhaps automatically drawing diagrams). See the top-level requestToBuy transaction, which declares that it invokes the processPurchase action on the manufacturer.

A *role* can be thought of as a kind of class or interface (this needs more thought) that captures relationships between participants. Likely, a participant will need to be able to fulfill more than one role. For example, ownership might be captured as a particular role.

This approach is sketched in the following listing.

```
role Dealing {
}

role Manufacturing { // gerund indicates role
  Shipper shipper;
  Inventory inventory;
}

participant Manufacturer fulfills Manufacturing {
  // Placeholder: track fulfilled orders

  action processOrder(Order o) {
    if (inventory.contains(o.car)) {
      Car@Owned car = inventory.remove(o.car);
      s.ship(car);
    }
    // Placeholder: record order in list of fulfilled orders
  }
}
```

```

}
}

participant AutoRipOff fulfills Dealing {
// ...
}

asset Car {
Manufacturing manufacturer;
// ...
}

asset Order {
int purchasePrice;
Car@Unowned car;
Dealing dealer;

Order(int p, Car@Unowned c, Dealing d) {
purchasePrice = p;
car = c;
dealer = d;
}
}

transaction requestToBuy (Car@Unowned car, int price)
returns Order // perhaps this should not return an order.
by AutoRipOff d // this implementation is specific to one dealer
issues processOrder to Manufacturer m
{
assert(car.manufacturer == m);
Order order = new Order(price, car, d)
manufacturer.processOrder(order);
return order;
}

```

### RequestToBuy: Obsidian example

This is an Obsidian [17] implementation of the requestToBuy scenario. As of this writing, it compiles in Obsidian.

It is likely possible to automatically infer a high-level process diagram similar to Figure 5. One main discrepancy is that Obsidian has no notion of high-level “process”; such a notion might need to be added so that sequences of transactions could be given appropriate names for the diagram.

```

contract Shipper {
...
transaction ship(Car@Owned >> Unowned car) {
// Placeholder: take ownership of the car.
// For now, throw it out :(
disown car;
}
}

contract Inventory {

```

```
...
transaction contains(Car@Unowned c) returns bool {
return true;
}

transaction remove(Car@Unowned c) returns Car@Owned {
// Placeholder: implement business logic here
return new Car();
}
}

contract Dealer {

transaction requestToBuy (Car@Unowned car, int price)
returns Order@Shared
{
//assert(car.manufacturer == m);
Order@Shared order = new Order(price, car, this);
car.manufacturer.requestToBuy(order);

return order;
}
}

contract Manufacturer {
Shipper@Shared shipper;
Inventory@Shared inventory;

transaction requestToBuy(Order@Shared o) {
if (inventory.contains(o.car)) {
Car@Owned car = inventory.remove(o.car);
shipper.ship(car);
}
}

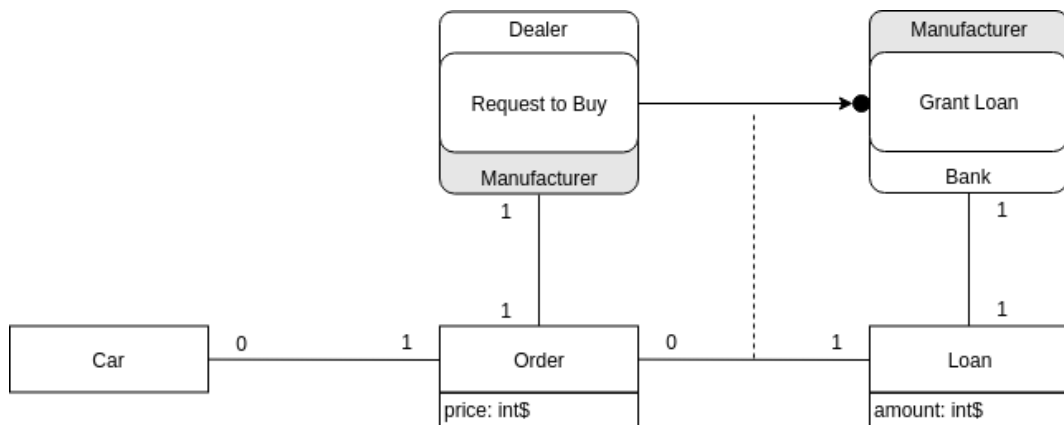
}

contract Car {
Manufacturer@Shared manufacturer;
}

contract Order {
int purchasePrice;
Car@Unowned car;
Dealer@Shared dealer;

Order@Owned(int p, Car@Unowned c, Dealer@Shared d) {
purchasePrice = p;
car = c;
dealer = d;
}
}

main contract ... { // Placeholder for main contract }
```



■ **Figure 9** The specification of logging for the loan grant.

#### 4.5.6 An Approach for Specifying Collaboration Logs

The exchange of information mediated by transactions on the blockchain necessitates a guidance towards the information that are going to be written in the ledger, from those that are meant to be kept off-chain. The information provided will serve not only as process documentation but also as a scheme to later extract the necessary information pertaining to process instances, should auditing be required. This becomes of particular significance to enable process mining on data recorded on the blockchain [60, 5].

Figure 9 illustrates an example using a notation inspired by the Object-centric Behavioral Constraints (OCBC) notation [61] and BPMN process choreographies [20]. In the example, we focus on the process fragment pertaining to activities *Request to Buy* and *Loan Money*. In the scheme, with  $\rightarrow\bullet$  we denote that the latter activity cannot take place before the former one, i.e., a *precedence* constraint holds true between records of the corresponding transactions. However, the loan is granted because of a specific sales order, therefore a connection lies not only at a control-flow level, but also as a data-level dependency. Therefore, we specify that the *Loan* refers to exactly one *Order*. Likewise, the *Order* is placed for a *Car*. The linkage of records in the transactional data will thus require that identifiers of the referred concepts be included in the dependent ones. Additional attributes can be specified along with their data type that are required to be registered in the transactions. In the example, we focus especially on values referring to amounts that the counterparts transfer as value attached to transactions, hence the \$ symbol added to the integer data type. In addition to the information pertaining data, transactions record the senders and recipients. We therefore enrich the model by adding them to the activities as in process choreographies. This will clarify the role covered by the accounts involved in the transaction. In the example, the *Dealer* is the initiator and *Manufacturer* the recipient of the transaction signifying the *Text to Buy* activity. Likewise, the *bank* is the sender of *Grant Loan* toward the *Manufacturer*.



## 4.6 Blockchain Data Analytics: Example of Decentralization of Service-Provider Platform

*Alevtina Dubovitskaya (EPFL – Lausanne, CH), Avigdor Gal (Technion – Haifa, IL), Stephan Haarmann (Hasso-Plattner-Institut – Potsdam, DE), Stefanie Rinderle-Ma (Universität Wien, AT), Francesca Zerbatò (University of Verona, IT)*

License © Creative Commons BY 3.0 Unported license  
© Alevtina Dubovitskaya, Avigdor Gal, Stephan Haarmann, Stefanie Rinderle-Ma, Francesca Zerbatò

Trust and reputation are the pillars, on which online marketplaces are built. Trust is often enabled by reputation [38]. In turn, reputation itself can be defined as *the collection of opinions received from other entities* [25]. Reputation is often used to frame the perception and expectation about someone’s behavior based on previous interactions (presumably similar to future ones). Therefore, reputation systems (being an example of collaborative sanctioning/filtering systems) are commonly used to build the trust and to facilitate transactions that happen in online marketplaces.

Many online marketplaces, e.g., eBay, Uber, or Airbnb, use *centralized* reputation management system. In such system, information about the performance of a given participant is collected as ratings from other members in the community who have had direct experience with that participant. The central authority that collects all the ratings then derives a reputation score for every participant, and makes all scores publicly available [30].

In such settings, trust is based not only on the content of the recommendations and rating provided by the platform users. In addition, trust is based on the credibility of the central authority that is solely responsible for ensuring integrity and authenticity of the evaluations provided by the users and for computing the reputation score, and thus enabling the trust. Therefore, an apparent drawback of a marketplace with central reputation-management entity is a single point of failure of the system: can be temporally unavailable or under control of the adversary <sup>7</sup>.

To eliminate the risk of having a single point of failure, a central authority that is responsible for reputation management can also be distributed. Indeed, there are environments where a distributed reputation system, i.e. without any centralized functions, is better suited than a centralized system. Distributed reputation systems rely on distributed communication protocol, and reputation computation method used by each individual. However, in case of orthogonal interests of the participants (such as “Hosts” and “Guests” of Airbnb platform), members of the platform can be assumed to be non-trusted, thus, we cannot just rely on them for computing the reputation score correctly. Emerging blockchain technology offers a way to execute processes in a trustworthy manner even in a network without any mutual trust between nodes [40].

Blockchain is a distributed technology that employs cryptographic primitives, and relies on a specific membership mechanism and consensus protocol [12] to maintain a shared, immutable, and transparent append-only register [31, 45]. Data, in the form of digitally signed transactions broadcasted by the participants, are grouped into blocks chronologically and time-stamped. A hash function, applied to the content of the block, forms a unique block identifier, which is stored in the subsequent block. A possible modification of the block content can be easily verified by hashing it again, and comparing it with the identifier from

<sup>7</sup> Due to the nature of the Internet, Airbnb cannot guarantee the continuous and uninterrupted availability and accessibility, <https://www.airbnb.com/terms>

the subsequent block. The blockchain is replicated and maintained by every participant. A malicious attempt to tamper the information stored in the registry will be noticed by the participants, thus guaranteeing immutability of the ledger. Smart contracts defined to execute arbitrary tasks, enable implementation of desired functionality on top of blockchain.

We can distinguish between *permissionless* and *permissioned* (*public* and *private*) blockchain systems. A system is permissionless when the identities of participants are either pseudonymous or anonymous [57], so that every user may participate in the consensus protocol, and, therefore, append a new block to the ledger. In contrast, in a permissioned blockchain identities of the users and rights to participate in the consensus (writing to the ledger and/or validating the transactions) are controlled by a membership service. A permissioned blockchain is *public* when anyone can read the ledger but only predefined set of users can participate in the consensus, and *private* when even the right to read ledger is controlled by the membership/identity service.

Employing the blockchain technology in the environment of non-trusted or competing entities that rely on reputation management can bring the following benefits. Blockchain by the means of smart contracts can provide transparency and credibility to the way the reputation score is calculated. In addition, the distributed ledger will guarantee immutability and availability of the history of ratings, and reputation scores. However, applying blockchain technology is not straightforward, especially in cases when the data flowing in the system have different levels of sensitivity, volumes, and dynamicity. Moreover, depending on the choice of the blockchain technology implementation, different data-management challenges may need to be addressed.

There already exist an ongoing effort for creating a distributed platform<sup>8</sup> using blockchain technology to provide similar services as the Airbnb platform. Yet, the following question remains: can a decentralized ledger substitute the role Airbnb plays in bringing hosts and guests together? [48] The goal of this work is to analyze different approaches of applying the blockchain technology in the real-world settings, using the example of online marketplace that employs reputation management, i.e., Airbnb. We would like to examine in detail the challenges that arise when permissioned and permissionless blockchain technology implementations are used. Then we propose potential approaches to address the identified challenges, and draw the directions for the future research in the area of blockchain data-management for reputation-based systems.

#### 4.6.1 Ensuring required properties of reputation-management systems using blockchain

In this section, we first provide an overview of Airbnb platform, which is used as an example of an online service-provider platform that relies on centralized reputation management approach. We reason about advantages of decentralizing such service-provider platform. Then, we list the required properties of a reputation-management system, and propose the approaches to ensure them using blockchain.

The Airbnb platform is an online marketplace that enables registered users (“Members”) and certain third parties who offer services (“Hosts”) to publish such Host Services on the Airbnb platform (“Listings”) and to communicate and transact directly with Members that are seeking to book such Host Services (Members using Host Services are “Guests”). Host Services may include the offering of accommodations and a variety of other travel and non-travel related services.<sup>9</sup>

---

<sup>8</sup> <https://www.beetoken.com>

<sup>9</sup> <https://www.airbnb.com/terms>

Airbnb platform, as a service provider, claims no responsibility regarding different kinds of situations that may happen. The Airbnb service can also be temporally unavailable and does not control or guarantee “*the existence, quality, safety, suitability, or legality of any Listings or Host Services, (ii) the truth or accuracy of any Listing descriptions, Ratings, Reviews, or other Member Content..., or (iii) the performance or conduct of any Member or third party*”. In the terms of service, it is stated that the platform is independent, meaning that no endorsements are provided to any of the registered users and the users only may receive some help in facilitation of the resolutions of disputes.<sup>9</sup>

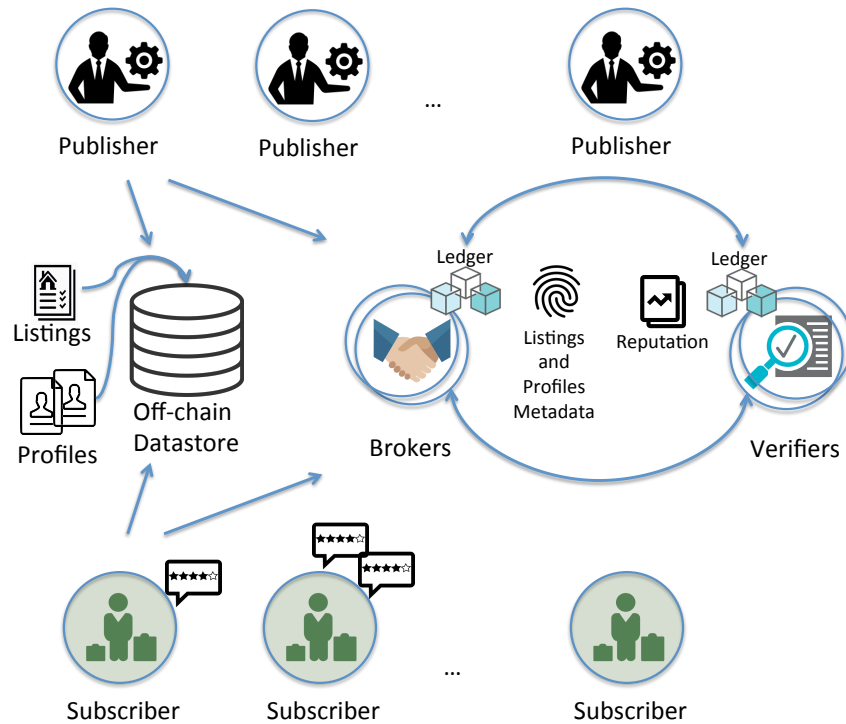
While this centralized and neutral approach facilitates the management of the online marketplace, it can cause issues such as fake listings (and web-pages duplicating genuine Airbnb web-page) created with non-existing properties, problems with local regulations, fake reviews, non-existing users, and non-longevity of the user identity, or listings [33]. The last three issues can sabotage trust derived from the reputation-management system relying on ratings and reviews. For such a centralized platform, it is probably almost impossible to provide the control over and guarantees regarding the behaviour of users, and the quality of the listings provided, due to the highly distributed geographic location of users in countries with different laws and regulations. Decentralization of the platform already could possibly improve the current situation by capturing specifics of the geographic regions, by involving users on a local level to contribute to verification of genuineness of user’s identity, listings, and ratings. This will lead to a correct functioning of the reputation-management mechanisms and the increase of transparency and trust.

Resnick et al. [52] claim that reputation systems must have the following properties: (i) Entities must be long lived: it should be impossible or difficult for an entity to change identity or pseudonym and erase the connection to its past behaviour; (ii) Ratings about current interactions are captured and distributed; (iii) Ratings about past interactions must guide decisions about current interactions.

The first property is difficult to achieve in an online marketplace. It is hard to control digital identities unless the binding to the real certificate/ID exists. The latter could be provided after verification using local administration databases, and storing such information in an immutable manner. Second property required availability of the entity/entities that capture and maintain the ratings. While centralized approach may fail due to (temporal) unavailability, in distributed settings the challenge is ensuring consistency of the data among distributed entities. In addition, the second property also depends on the willingness of participants to provide ratings, for which there must be some form of incentive. The third property depends on the usability of reputation system, and how people and systems respond to it [30].

Based on the identified need to improve trust management, and taking into account required properties of reputation systems, we propose to employ the blockchain technology for the following purposes:

- (i) verifying existence of the user identities and listings: for example, when booking a property, the user wishes to verify it exists and that the offer is valid;
- (ii) validating the quality of the the unit (apartment or service) and its correspondence to the information provided in the listing (reputation of the user can also be used to determine different user roles in the network: more reputable users have more “stake in the network”, thus will not be willing to sabotage the system);
- (iii) ensuring transparency in communications and credibility of the rating-management system;
- (iv) improving the conflict resolution process by, e.g., providing guarantees regarding user payments: smart contract can be used as escrow for safeguarding the payment and setting the terms of payments (the terms can be negotiated).



■ **Figure 10** System Model.

#### 4.6.2 Decentralization of a service-provider: system model and design goals

Following the organization of Airbnb service-platform, we consider *Publishers*, with the “Host”-role, and *Subscribers* of the platform, with the “Guests”-role. Publishers are the owners of the property or providers of a service that they want to advertise on the platform, and Subscribers are the renters of the property or the users of the services. In addition, we define the following stakeholders: *Brokers* – entities that may be used as the intermediary between Publishers and Subscribers, and *Verifiers* of the content of the published information (“Listings”), and the correctness of the data (and their correspondence to the physical world). For a user to become a Verifier, a certain level of reputation is required. Verifier can also use external trusted database such as the database of the housing properties registered in the local administration. Publishers and subscribers have orthogonal interests. Hence, neither of them can be fully trusted. Moreover, there is competitiveness within each group: subscribers are looking for the best available listing, and publishers are willing to rent a property or provide a service to the most reliable subscriber.

Next, we list the design goals (*DG*) regarding the functionality of a blockchain-based distributed service-provider platform. The following actions are required to be available for the users by the platform:

- *DG-1*: registration of a user, ensuring longevity of the user identifier, and preventing from leaving the network and joining with another;
- *DG-2*: registration/removal of a service, or a property to rent out (posting a listing);
- *DG-3*: searching for a service or a unit to be rented;
- *DG-4*: verifying existence of a property of service;

- *DG-5*: booking (negotiation regarding renting conditions such as payment mode, precise timings of arrival, transfer of the key, confirmation, cancellation);
- *DG-6*: payment settlement (with or without involving cryptocurrency, but with respect to the agreement reached during negotiation);
- *DG-7*: providing evaluations and ratings (for publisher by subscriber and vice versa) and computing reputation;
- *DG-8*: detection of the collusion between users and malicious behavior, conflict resolution.

Trying to attain these goals we will rely on the following properties of the blockchain technology. Immutability and append-only properties can be leveraged (*i*) to ensure the history of all the registrations and ratings, and (*ii*) to keep track of the updates of the listings, which can be performed via smart-contract functionality. Smart contracts will be used as well for transparent computation of reputation (based on the provided ratings), for negotiation of the terms of booking, and settling cancellation. Potentially, the full history of renting, ratings, and communications between users (for instance, when negotiating booking conditions) can be leveraged using machine learning approaches to assist subscribers searching for a unit or a service and to detect the malicious behavior trends.

### 4.6.3 Challenges when applying blockchain

While the advantages of employing the blockchain technology listed above advocate for its adoption, in order to achieve the design goals, the following challenges have to be addressed first.

- *C-1: Choice of the blockchain-technology implementation.* The choice of technology highly depends on the use-case scenarios. It is crucial to define the mapping between the “peers” and the real-world entities and to define who will maintain the blockchain, i.e., who are the peers/entities that will be storing a distributed ledger? Depending on the sensitivity level of the data that flow in the system, as well as the degree of the involvement of the peers, next step is setting up the policy to join the network and to read/write the transactions from/to the ledger.
- *C-2: On- and off-chain data management.* It is highly important to define what kinds of data will be stored on- and off- blockchain to avoid unnecessary data replication that can make a system impossible to use in real-world settings due to high latency, privacy issues, and requirements to process big volumes of data. Definition of the structure and formats of on- and off-chain data depends on how the following challenges are addressed:
  - *C-2a: Translating business processes and negotiation terms to smart contracts.* Desired functionality of the system and capabilities of the smart contracts to capture it have to be considered first, to understand what is the minimum required amount of the data that have to be stored on the distributed ledger. For instance, in the case of Airbnb-like platform, we assume a possibility for negotiation of booking conditions. Therefore, the terms to be negotiated have to be taken into account beforehand. Using smart contracts for automatic payment settlement requires financial data about the user to be available in the system and coordination with banks and companies providing online payment services.
  - *C-2b: Ledger design and data expressiveness.* The ledger is the list of append-only transaction, together with read-write set that is replicated among all the entities that maintain the ledger. Therefore, transactions organized in such a way are not efficient to query. The design of the read-write set (the data stored on the blockchain) has to ensure efficient queries and required functionality by defining the database structure

and data organization accordingly. Storing the whole data lake with all the listings provided by users and all the communications between Publishers and Subscribers is impractical, while keeping track of the users registration, or having metadata, ratings and evaluations is feasible.

- *C-2c: Data privacy and security.* The data stored on-blockchain and replicated among distributed entities can have different sensitivity levels. During user registration, as well as verification of user’s identity, or genuineness of the property or a service described in the listing, access to the highly sensitive data are required. The listing itself, once its validity is ensured, may contain only publicly-available data. Applying cryptographic techniques can become necessary to enforce access control policy and ensure data privacy and security. This, however, introduces the challenge of managing cryptographic keys during their whole life-cycle.
- *C-3: Consistency between digital representation of objects and the real-world.* Establishing and maintaining consistency between digital and physical worlds is challenging. Information about all the registered users/units/services has to be verified using some trusted sources, and the information about registered users and their reputation. With the absence of a centralized entity, it is also challenging to maintain consistency and handle conflict resolution.

The fundamental and first-to-address challenge is defining the choice of the technology and setting up the network and policies. Depending on the concrete use-case scenario and how *C-1* challenge is addressed, i.e., what type of blockchain technology implementation is chosen, different considerations have to be taken into account when resolving challenges *C-2* and *C-3*. We next discuss how the choice of blockchain technology influences challenges *C-2* and *C-3* and the potential approaches to address them.

#### 4.6.4 How to address data-management challenges when implementing distributed blockchain-based service-provider

The choice of technology shapes further challenges related to the blockchain data management. First, we provide more details about differences between permissioned and permissionless blockchain technology implementations. Then, we discuss research directions and potential approaches to address the aforementioned challenges *C-2* and *C-3* in the framework of applying permissioned and permissionless blockchain technology.

**Blockchain technology implementations:** Below we briefly describe the characteristics of the permissionless and permissioned blockchain technologies using their existing implementations with smart contract functionality as an example.

*Ethereum*[11] is an implementation of a permissionless programmable blockchain that enables any user to create and execute the code of arbitrary algorithmic complexity on the Ethereum platform: Ethereum Virtual Machine (EVM). EVM can be seen as a large decentralized computer. “Accounts” of two types could be created on EVM. Externally owned account (EOA) is an account controlled only by a private key of a user. The owner of the private key associated with the EOA can remain anonymous (up to a certain degree) and has the ability to send messages. Contract account is the second type of accounts that can be seen as an autonomous agent that lives in the Ethereum execution environment and is controlled by its contract code: smart contract. Smart contract is used to encode arbitrary state transition functions, allowing users to create systems with different functionalities by transforming the logic of the system into the code. In case of public blockchain (such as Ethereum Mainnet), smart contracts and all the transactions are public.

In Ethereum, transaction processing is Turing-complete and it can be used to implement any public functionality in a distributed fashion, but the code execution must be paid. The transaction price limits the number of computational steps for the code execution in order to prevent infinite loops or other computational wastage. Users can participate in the consensus process to obtain the tokens in order to pay for the transaction execution. In Ethereum, the consensus is achieved by using GHOST – modified proof-of-work (PoW) mechanism.<sup>10</sup>

In order to avoid issues of network abuse, all programmable computations in Ethereum are subject to fees. The fee schedule is specified in units of gas. Thus any given fragment of programmable computation (i.e., creating contracts, making message calls, utilizing and accessing account storage, and executing operations on the virtual machine) has a universally agreed cost in terms of gas [66]. Ethereum provides excellent scalability in terms of number of nodes and clients, but has a limited transaction throughput. In 2016, typical Ethereum throughput was fewer than 20,000 transactions per day, i.e., about 0.2 tx/s on average [63].

In contrast, due to the architectural design and different type of consensus protocols employed, scalability in terms of number of nodes in case of permissioned blockchain technology is limited. *Hyperledger Fabric* [24, 2] – an implementation of a permissioned blockchain – is an open source blockchain initiative hosted by the Linux Foundation. Hyperledger Fabric contains a security infrastructure for authentication and authorization (membership service). It supports enrollment and transaction authorization of peers and users through public-key certificates. This is one of the main differences with the permissionless blockchain framework. In Hyperledger Fabric, in addition to the membership service, the other main architectural components are peers, and ordering-service node, or orderer. Orderer is a node running the communication service that implements a delivery guarantee, such as atomic or total order broadcast. The ordering service can be implemented in different ways, ranging from a centralized service to distributed protocols that target different network and node fault models.

Architecture design of permissioned blockchain technology may introduce a certain level of centralization due to relying on the membership service and orderer (that, can also be distributed to prevent a single point of failure in the system). However, such a design provides privacy and security guarantees that are impossible to achieve in the permissionless settings.

Smart contracts are implemented by the chaincode that consist of Logic and associated World state (State). Logic of the chaincode is a set of rules that define how the transactions will be executed and how the State will change. The Logic can be written using general-purpose programming language. The State is a database that stores the information in a form of key-value pairs, where the value is an arbitrary byte array. The State also contains the block number to which it corresponds. The ledger manages the blockchain by including an efficiently cryptographic hash of the State when appending a block. This enables efficient synchronization if a node was temporary off-line, minimizing the amount of stored data at the node.

**On- and off-chain data management and consistency between the ledger and the real world:** In both settings, given the data volumes managed by the service-provider platform in an online marketplace, it is impossible to guarantee data privacy, consistency and efficient queries over the data, while keeping all the data on-chain. Innovative approaches for storing, indexing and describing the data are thus required to ensure privacy, consistency and efficiency. However, we could also leverage the possibility to store the data off-chain and the

---

<sup>10</sup> <http://www.ethdocs.org>

design and structure of the ledger to achieve aforementioned properties. Next, we discuss how could we address the challenges of on- and off-chain data management and achieving consistency between the ledger and the real-world within permissionless and permissioned settings.

*Permissionless settings.* Based on the characteristics and properties of the permissionless blockchain, we would like to point out several research directions and potential approaches that could be applied regarding on-/off- chain data-management challenge:

- Everyone can join the network, as there is no centralized entity, the mechanisms embedded in the smart contracts have to be developed to make sure that users identifiers are static. Pseudonymization approach could be used: such as creating multiple pseudonyms to ensure privacy, but that can be linked when computing reputation.
- Currently, PoW is used: transaction fees are involved. Therefore, we have to take this into account when writing smart contracts, as every operation have to be paid.
- Sensitive data, such as exact locations and names of the guests, should not be placed in plain sight on the ledger due to privacy issues. Therefore, there is a challenge in finding a way to perform operations over encrypted data (lightweight homomorphic encryption), or over statistical (aggregated anonymized) data.

*Permissioned settings.* For the case of permissioned settings we make the following considerations:

- The policy of joining a network and differentiating users' rights into those who can add new blocks, submit a transactions, read the ledger, *etc.*. A possible approach is to let only users with the certain level of reputation to maintain the ledger. However, incentive mechanisms and rewards have to be defined.
- In general, in case of permissioned blockchain less peers will be maintaining the ledger (data replication is simplified, however, still depends on the consensus mechanism employed). All of them will also be registered. Therefore, access control policy can be partially enforced already by membership service. Implementation of the cryptographic techniques in the distributed environment with less peers is more practical, yet further investigation are necessary.
- Membership service could also be employed to ensure longevity of the user's identifier. Yet the exact requirements for constructing pseudonyms/identifiers have to be developed.
- Digitization of business processes (e.g., verification and publish the listing) can be done via smart contracts and secured with endorsement policy (such as in the case of consensus protocol employed in Hyperledger Fabric v1.X [2] ). For instance, verification of a listing will require the endorsements from (i) trusted sources, such as government registry and (ii) a number of local peers with a certain level of reputation.

**Consistency between the ledger and the real world:** The *C-3* challenge related to ensuring that the ledger stores genuine information is very hard to achieve, in both settings. Smart contract functionality can be employed to automatize the process of verification of the information using trusted databases, or the local peers with a certain level of reputation. For such verification to be reliable, the system has to ensure collusion detection and provide a possibility to exclude the malicious/colluding peers. In permissioned settings, for instance, this can be achieved through the policy defined on the side of the membership service.



#### 4.6.5 Conclusion

The expansion of blockchain-based applications proliferates in different areas, and online marketplaces are not an exception. Centralized service providers, especially the ones relying on reputation and trust, will definitely benefit from adopting the principles of the blockchain technology, and the properties it brings, namely transparency, immutability, and credibility, to name a few. Yet, multiple data management challenges arise when such decentralization occurs.

In this work, we analyzed these challenges using a practical potential use-case scenario of Airbnb service-provider platform. We proposed possible ways to address these challenges in different permissionless and permissioned settings, indicating the directions for the future research in the area of blockchain data-management for reputation-based systems.

Addressing the challenges only from the data-management perspective may not be enough. However, it could simplify compliance of such applications with the local laws and regulations. Intelligent data management, together with transparency and credibility brought by the blockchain, opens the doors for leveraging machine learning techniques to enhance and facilitate the use of the online marketplaces, for instance to enable recommendations of the unit or the service that can be of interest for a subscriber.

### 4.7 Data Technology to the Rescue: Digging the D in GDPR

*Søren Debois (IT University of Copenhagen, DK), Alevtina Dubovitskaya (EPFL – Lausanne, CH), Avigdor Gal (Technion – Haifa, IL), Petr Novotny (IBM TJ Watson Research Center – Yorktown Heights, US), Stefanie Rinderle-Ma (Universität Wien, AT), Stefan Schulte (TU Wien, AT), Ludwig Stage (Tübingen, DE), Kaiwen Zhang (ETS – Montreal, CA)*

The motivation for the discussion presented in this section is the recently introduced GDPR (General Data Protection Regulation) 2016/697, an EU regulation that aims at regulating the way personally identified data is being gathered and consumed and to define the legal rights of people to the use of their data. Taking the point of view of computer and data scientists, we wish to identify suitable mechanisms to support organizations in their journey towards the compliance of their information systems with GDPR.

The discussion involved a deeper understanding of the GDPR and the requirement it puts with respect to data protection. Equipped with this understanding, we have identified elements of matured (databases) and new (blockchain) technologies that can be put into use when adapting an organization's information system to be GDPR compliant.

#### 4.7.1 GDPR 101

The EU General Data Protection Regulation [1] came into force May 25, 2018. It applies within the EU and the European Economic Area; however, because information-centric businesses tend to be global, it is of global concern.

The GDPR confers onto citizens (*data subjects*) a number of rights, and onto companies and institutions using that data (*data controllers*) a number of obligations, the latter at the penalty of potentially significant fines.

Of particular interest to this chapter are the following stipulations of the GDPR. It is worth noting that our discussion is not comprehensive.

- GDPR is concerned exclusively with personally identifiable data. It specifically states that anonymisation stops data from being personally identifiable. Pseudonymisation does not, but is considered, in some cases, an adequate security measure [43].
- Any processing of data must be for a specific purpose, and that purpose must be legitimate. Common legitimate purposes include those that are necessary towards fulfillment of a contract (*e.g.*, I must record your address to ship you goods) as well as those required by law, those required for ongoing court cases, *etc.*
- Data subjects have a number of rights:
  - The right to erasure (Article 17): When a purpose ceases to be legitimate, perhaps because it has run its course, or perhaps by request from the data subject, a data controller must erase **without undue delay** personal data.
  - The right to rectification (Article 16): When given notice of incorrect data, a controller must, again **without undue delay** rectify incorrect data.
  - The right to data portability (Article 20): Upon request, a data subject may receive from a controller all of his/hers personal information in electronic form.
- GDPR imposes a number of obligations on data controllers. The group discussed the following:
  - The obligation to keep records of processing activities (Article 30).
  - The obligation to inform data subjects about processing (Article 13, 14)
  - The distinction between the data controller and its data processors, and the requirement that a data controller has a contractual agreement with its processors.

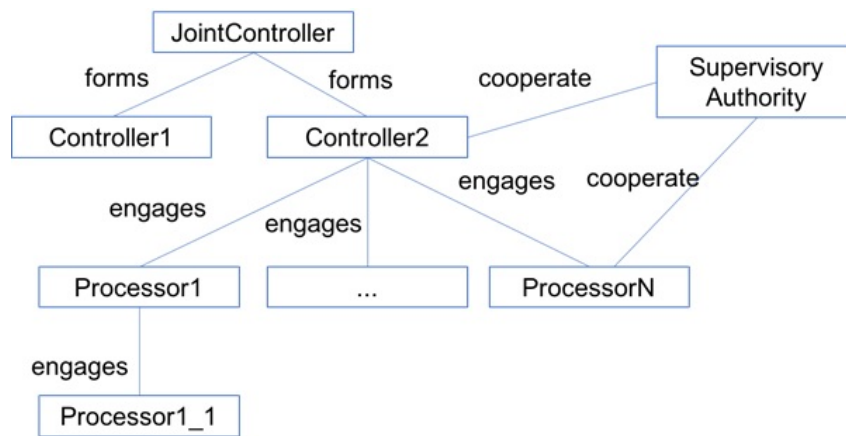
### Controllers and Data Processors

The GDPR poses many challenges on interoperability between partners exchanging data. Partners can basically have two roles as defined in Article 4 of the GDPR: a controller “which [...] determines the purposes and means of the processing of personal data”<sup>11</sup> and a processor that “processes personal data on behalf of the controller”. In addition, there might be a supervisory authority that monitors the compliance with GDPR in an EU member state. The controller and the processor(s) might have to cooperate with the supervisory authority. It is also possible that several controllers jointly determine the purpose of the processing, then they are called joint controllers. In agreement with the (joint) controller the processors can engage other processors.

Figure 11 shows a potential network of partners. All engagements in such a network *e.g.*, between controller and processor) are subject to legally binding contracts with respect to the GDPR and thus in essence with respect to the use of the data. Article 30 states that the controller “shall maintain a record of processing activities under its responsibility” and the processor “shall maintain a record of all categories of processing activities carried out on behalf of a controller”. Already these two basic obligations (establishing contracts and logging) for data exchange under GDPR point to the usage of blockchain technology due to its support for decentralization, trust, and transparency. Firstly, the legally binding contracts between controllers and processors, and possibly also between joint controllers and processors that engage other processors, could be possibly implemented as smart contracts. Secondly, the corresponding logging of the different partners (with different roles) could be realized using blockchain technology. Despite the aforementioned potential benefits, details of the realization and the resulting complexity are yet to be investigated.

---

<sup>11</sup> <https://advisera.com/eugdpracademy/gdpr/definitions/>



■ **Figure 11** Partner interoperability

#### 4.7.2 Technological Background: Data Life Cycle, Databases, and Blockchains

The data science discipline defines a lifecycle of data that captures roughly four steps: data is first **gathered** using possibly sensors, human input, or otherwise already available in data stores (such as the Web). The gathered data is **managed**, integrated, and stored on a facility that may range from a personal storage to a cloud storage, by means of data management systems such as a database management system (DBMS – see below). The stored data is prepared to be **analyzed** by machine learning algorithms and the outcome is finally **presented** to users by means of database queries, managerial dashboards, *etc.*

Database technology has been around for many years now (*e.g.*, [51]). Using a DBMS, one can define a *schema*, describing the content of the database using a conceptual model such as the relational model. Data is accessed via *queries*, which can either be retrieval queries or update queries (insertion, modification, and deletion). A *log* of all update activities is kept to assist in situations of crash recovery or other failures. Finally, data may be distributed over multiple sites and partially replicated to ensure speedy retrieval and to guard against failures.

The emergence of blockchain technology has opened manifold opportunities to redesign collaboration. In general, blockchains allow to store data in a distributed way, with each participant in a blockchain network being able to possess the complete blockchain (allowing transparent data sourcing) and verify the stored data. Blockchain technology does not rely on trusted third-party architecture, which creates entry barriers and a single point of failure. Rather, blockchains guarantee the integrity of the data and smart contracts enable distributed execution without delegating trust to central authorities nor requiring mutual trust between each pair of parties. Furthermore, blockchain technology potentially enables fine-grained access controls, allowing different parties to selectively share different data with different partners, using data that is resident in the blockchain.

Three specific elements of blockchain are important to the following discussion. First, blockchain can serve as a trusted (ledger) data storage. No data, once written to the blockchain, can be modified without letting everybody with access to the blockchain knowing about it. Second, the use of smart contracts, computer scripts that define a sequence of reactive operations on which partners agree ahead of time, enable a transparent operation of

a system, subject to the audit of all participants. Finally, blockchain employs mechanisms for replication of the data on a blockchain that allow the continuity of recording activities even when faced with network instability.

There is a common distinction between *permissionless* and *permissioned* (*public* and *private*) blockchain systems. A system is permissionless when the identities of participants are either pseudonymous or anonymous [57], so that every user may participate in the consensus protocol, and, therefore, append a new block to the ledger. In contrast, in a permissioned blockchain identities of the users and rights to participate in the consensus (writing to the ledger and/or validating the transactions) are controlled by a membership service. A permissioned blockchain is *public* when anyone can read the ledger but only predefined set of users can participate in the consensus, and *private* when even the right to read ledger is controlled by the membership/identity service. In this work, we refrain from discussing the various design options that come with blockchain system being permissionless or permissioned.

In public networks like Bitcoin and Ethereum, storage of large amounts of data is expensive. Therefore, there is a need to rely on off-chain file sharing networks, such as IPFS [7], Filecoin,<sup>12</sup> and Swarm.<sup>13</sup> The basic functionality is to record a hash of the document on the blockchain, send the original document on the file sharing network, and retrieve the document using the content address which was previously stored on the blockchain.

In IPFS, there is an assumption that participating nodes are altruistic, and are willing to store data simply because they wish to maintain the network available. However, this altruistic model is weakly applicable in practice, with files usually remaining available for 24 hours at most. To obtain longer availability, users rely on pinning services, which will actively maintain the information long-term, in exchange for financial compensation.

To address these issues, newer systems like FileCoin and Swarm have built-in incentive mechanisms to promote long-term file availability.<sup>14</sup> In FileCoin, participating nodes employ Proof-of-Storage to mine blocks and collect mining rewards. Mining power is calculated based on the amount of data stored, which can be verified using Proof-of-Retrievability. In Swarm, positive incentivization is provided when serving content, which incentivizes nodes to retain popular files. In addition, negative incentivization is enforced through a staking (security deposit) mechanism, where nodes have to commit some cryptocurrency resources in order to participate. An audit mechanism then allows the data owner to challenge storage nodes, who must provide the requested data, or have its stake slashed.

### 4.7.3 Personally Identified Data

Putting personally identifiable information on a public blockchain in an immediately readable (*i.e.*, non-encrypted) form is likely prohibited specifically because (i) a blockchain, as an immutable database, can never satisfy the right to erasure, and (ii) because it would be impractical for a data controller to obtain a contract with every node in the blockchain as a data processor. These limitations may be resolved when the data are stored and or modified such that the personal identifying information is not present. Moreover, the willingness of individuals to agree to the data collection can be increased with such a solution.

---

<sup>12</sup> <https://filecoin.io/>

<sup>13</sup> <https://swarm-guide.readthedocs.io/en/latest/introduction.html>

<sup>14</sup> <http://swarm-gateways.net/bzz:/theswarm.eth/ethersphere/orange-papers/1/sw%5E3.pdf>

Tokenization is a technique, aimed at separating the true identity of a person and its representation in a database.<sup>15</sup> Tokenization has two main variations, namely anonymization and pseudonymization. The use of tokenization mechanisms allows to disconnect the various types of data about a person (such as income, education, *etc.*) from the data uniquely identifying the person (such as social security number). The mapping information between the various types of data is separated from the personal identifying data and stored within a secure storage of the tokenization system. This approach allows the use of personal data in data processing tasks without revealing the actual identity.

Tokenization may allow re-identification [55] of personal data from tokenized data, especially if the same pseudonym is used repeatedly for the same person. Successful re-identification does not expose personal data directly since there is no personal data stored on the blockchain. What would be leaked is information about what kind of data is stored for a particular person and for which purposes, which may, by itself create a breach of GDPR regulations. To provide controlled linkability, Camenisch and Lehmann proposed a combined approach of pseudonymization with a potentially untrusted server that stores the mapping [13]. Pseudonyms should be unlinkable by default, yet preserve the correlation that enables to re-establish the linkage only if necessary and based on the policy specified via a (potentially untrusted) converter. The converter establishes individual pseudonyms for each server derived from a unique main identifier that every user has, but without learning the derived pseudonyms. The converter is still the only authority that can link different pseudonyms together, but it does not learn the particular user or pseudonym for which such a translation is requested. To construct such framework, the authors use dual-mode signatures (which allow one to sign messages in the plain as well as when they are contained in an encryption), (verifiable) pseudorandom functions, and homomorphic encryption.

In case of a unique mapping, all historical activity can be traced, which is both advantageous and undesirable at times, for example when it comes to preserving privacy. In medical studies, double-blind studies require that even the medical personnel should not know who is actually provided with a novel medical treatment and who is part of the control group. AnonRep [69] is the first practical anonymous reputation system maintaining the unlinkability and anonymity of users' historical activities. AnonRep uses verifiable shuffles and linkable ring signatures, with a multi-provider deployment architecture.

Similar to the right of data erasure, data subjects can request that only anonymized data can be used for the data analysis tasks. This is also somehow beneficial for the data controllers: if the data are anonymized, there is no need to comply with GDPR.

How can we actually guarantee that the data are properly anonymized and re-identification of the data subject is impossible, taking into account that more data about the same data subject can be revealed in the future? Depending on the nature and sensitivity of the data, and the field of the research question for which the data are used, there are various considerations such as what is a required privacy level and how to choose the parameters of the anonymization algorithms to achieve this level of data anonymization (*e.g.*, *k*-anonymity). Blockchain technology can be used in order to track the data that have been released in the anonymized form, including the anonymity level of the data (but not necessary keeping all the data, only some metadata). This will be used as an input to compute the risk of being re-identified if more data about the data subject are released. Kiyomoto *et al.* [32] propose a blockchain-based distribution scheme for anonymized datasets. The platform consists of peers that act as data brokers, data receivers, and verifiers of transactions, and

---

<sup>15</sup> [https://en.wikipedia.org/wiki/Tokenization\\_\(data\\_security\)](https://en.wikipedia.org/wiki/Tokenization_(data_security))

blockchain is used for recording all transactions of anonymized datasets between a data broker to a data receiver. One can argue that keeping track of all the transaction can put the anonymization at risk. However, as mentioned above, recording of all the transactions can be used to compute the risk and prevent the violation of the data subject's privacy.

Storing access control policy/permissions and transaction history on the blockchain has the benefit of letting the user transparently identify who has/had access to her data and for which purpose. Sensitive data may also be stored on blockchain, but in this case the data must be encrypted, key management and access control mechanisms must be put in place (for instance, via smart contracts).

#### 4.7.4 Data Gathering and Usage

Within the GDPR, data can be processed, including being gathered, only for a legitimate purpose, to be clearly defined in advance. Given a purpose and the necessary permissions, a DBMS may be designed, creating a database schema according to which data is collected using database insertion and update queries.

Recall that a database log details the sequence of activities (insertions, updates, and deletions) performed by the database. The append-only character and the property of immutability of blockchains make them suitable for logging purposes. Logging only needs to append data, and logging an update is simply done by creating a new log entry. The immutability feature might be beneficial in terms of creating a tamper-free audit trail that could establish a higher level trustworthiness in an audit report done with the help of such a log.

A log responsibility can be extended to serve as a mechanism for ensuring that data is used solely for its legitimate purpose by using a blockchain to store the log. The extended log (performed using an API/gateway/wrapper) will include all the queries that were performed throughout history on the database. Its append-only nature, combined with mechanisms to avoid tampering with the registered data, allows a trustworthy mechanism to record database queries for possibly future auditing and other tasks that are relevant to GDPR.

When using blockchain to store a DBMS log in a trustworthy fashion, three main questions come to mind. First, how can one trust the controller or any of its contractual data processors to faithfully use this mechanism? This is, in fact, a question that is beyond the scope of GDPR. We note here that GDPR is meant to define the procedures that are needed to be set in place to ensure the privacy of personally identifiable data. The enforcement of such mechanisms, once defined to be in place, is left to be performed by other conventional means already specified by law.

Second, how can one guarantee the correctness of the log in the face of database failures. For this, a rich literature exists, on the use of a *write ahead log* to allow safe recording of database activities.

Third, how can one ensure the correct recording when it comes to blockchain underlying mechanisms. In particular, how can one ensure that blocks containing relevant log information will not be discarded. This can be possibly done by using an incentive mechanism, inherent to blockchain, which ensures a positive payment for storage.

GDPR internal and external auditing can be done using the blockchain-based storage and the software systems, thus certifying that a company has put the technical means in place in order to comply with GDPR requirements.

#### 4.7.5 Right to Erasure

The GDPR provisions for the erasure of personal data upon request. Any mechanism that supports the right to erasure should also be able to provide a positive proof to the erasure that will serve for auditing purposes.

Data management involves a set of steps that are aimed to ensure the storage of a semantically meaningful data. This is typically done by the use of a DBMS, which capabilities provide semantic guarantees over the data. One such guarantee is the maintenance of integrity constraints, allowing to maintain a connection among various elements of data. When it comes to the Right to Erasure, such a request is translated into a delete query in the database. The query is duly reported in the log (now stored on blockchain), which can be served as a proof of compliance with the request. A point to keep in mind is that whenever required by law (*e.g.*, for tax purposes or money laundering regulations) data cannot be erased in response to a request for erasure. In such a case, the database consistency may be infringed. Such situations can be handled by advanced techniques for exception handling in database consistency.

Data in databases may be distributed (using distributed database techniques) and partially replicated on a peer system (using methods that are similar to IPFS). Such an architecture requires specialized mechanisms to deal with erasures on multiple peers. This can be possibly done by using an incentive mechanism, where data that is intended for erasure will no longer receive positive payment for storage and will eventually be dropped from the peer network. We note that such a mechanism is in line with GDPR, which requires that the request for erasure will be performed without undue delay, rather than immediately.

The use of blockchain technology for storage of personal data thus must consider the implications of storing data onto an immutable ledger and employ appropriate mechanisms to allow the erasure of personal data. Blockchain technology supports solely an append-only mechanism for storing data. This presents a challenge when dealing with the right to erasure, since all queries are recorded in the log. The use of sophisticated tokenization can assist in upholding the right to erasure. When using anonymization, the identification of a person cannot be recovered. When using pseudonymization, simply erasing the link between the pseudonym and the identifier can do the trick and turn the pseudonym into an anonymized entity, no longer traceable to the person it represents. In addition, there are several known approaches of effective erasure of data stored on the ledger that uphold the integrity of the ledger.

A related topic to the right to erasure is the enabling of an opt-out option from data gathering, even if the data is anonymized. If the data subject agrees to provide anonymized data, but would like to be informed about the use of the data (*e.g.*, the outcomes of a research study), the combination of pseudonymisation and anonymization can be used. If the data about the data subject were used as an input to an algorithm, then a data subject can require to stop processing the data. In such a case, a careful analysis is needed to check a possible impact on the privacy of other data subjects, whose data are processed by the same algorithm and stored in the same database.

#### Erasure mechanisms for tokens

Proof-of-Burn is a consensus mechanism for Proof-of-Work and Proof-of-Stake, proposed for public cryptocurrency blockchain systems [49]. In this mechanism, coins are burnt by sending them to an unredeemable output. In Bitcoin, this is accomplished by specifying

an output script that will never evaluate to true (*e.g.*, push 4, check if it is equal to  $5^{16}$ ). Once coins are burnt, a consensus algorithm that functions based on Proof-of-Burn relies on miners to supply the proof that sufficient coins have been burnt (*i.e.*, exceeds the required difficulty) to convince other miners to accept the block as valid. Because coins burning is sent as a regular transaction, it is necessary for the burning transaction to be stabilized (*i.e.*, accumulate sufficient confirmations). Hence, the consensus protocol would set a lower limit on the amount of confirmations necessary for valid proofs.

Proof-of-Burn can be used in the context of a GDPR-compliant public blockchain to support the right to erasure in the context of data tokenization. Given a blockchain platform where tokens have to be generated and redeemed for each piece of data in order to process this data further, a user who requests the right to erasure can notify the network to burn tokens associated to her account (per ID). Each account that currently owns tokens associated with this user must then submit a transaction to burn the tokens. After a stabilization period (measured in confirmations), the user can challenge any participant to demonstrate that her right has been executed. Challenged participants must then supply the appropriate Proof-of-Burn (*e.g.*, address to a transaction who spends the tokens, and sends them to an unredeemable output). Thus, the information embedded in the tokens are no longer usable, which stops further processing of this information.

This approach works insofar that each token output has a finite consumption limit attached to them (*e.g.*, transaction outputs for cryptocurrencies are single-use). For certain type of tokens, it is possible that their consumption is unlimited (the same transaction output can be used multiple times). To accommodate this case, the input script provided by the burning transaction should include a special operation (to be provided by the blockchain platform) that consumes the token permanently, barring future transactions from redeeming it.

One possible issue is that the burning transactions are never accepted into the blockchain, possibly due to low transaction fees attached, or blacklisting policies from certain miners that preclude the transactions from being included. Because unconfirmed transactions will not prevent the tokens from being processed further, it is the responsibility of the data controllers to ensure that the transactions are confirmed, whether this means that the transaction fees have to be raised, or that there is a sufficient proportion of miners who are willing to accept the transactions so that they will eventually be included in a mined block. This could be incentivized by adding special mining rewards, embedded in the blockchain core protocol, for blocks that contained burned tokens. This type of reward, coupled with the special burning operation detailed in the previous paragraph, could be used to demonstrate that a specific blockchain platform is GDPR-compliant.

Proof-of-Burn can also be useful for interoperability between multiple blockchains. In order to transfer a token from one blockchain to another, it must first be burned in the original blockchain. Once this burn transaction has been confirmed and stabilized, a new transaction on the new chain can be used to redeem the same data token, by supplying the Proof-of-Burn referencing the previous transaction. This ensures that the same token is not “double-spent” on multiple blockchains. This concept is already used for cryptocurrencies, when transitioning after a hard fork.

We note here that UltraNote provides self-destructing data storage (based on an expiry time) [59]. However, it is unclear if the underlying cryptomechanisms are usable when erasure is explicitly requested at an arbitrary time.

---

<sup>16</sup> [https://en.bitcoin.it/wiki/Proof\\_of\\_burn](https://en.bitcoin.it/wiki/Proof_of_burn)



### Cryptoeconomic erasure for off-chain storage

With a cryptographic approach based on either symmetrical or asymmetrical cyphers, the personal data are encrypted prior to storage onto the ledger. Later, when the data are retrieved from the ledger, the data must be first decrypted to reconstruct the contained information. The data become effectively erased when the decryption key(s) are not available. This approach requires an encryption and decryption key management mechanism, which upon request from the data owner or data manager erases decryption keys and thus erases the data encrypted with these keys. The use of homomorphic encryption allows a limited number of operations on already erased data while providing compliance with the GDPR requirements. The key management mechanism must allow encrypting the various combinations of data in logical units such that the later removal of the decryption keys leads to erasure of the intended (and only intended) data. Moreover, the key management mechanism must be trusted by all participants of the blockchain network and thus appropriate architectural and operational guarantees and policies must be put in place. For example, to provide an independent key management mechanism in a private blockchain network of equal participants, the participants may agree and elect an independent and trusted third party (centralized or distributed) responsible for the key management. It is worth noting that the cryptographic approach, if implemented correctly, provides guarantee of complete data erasure while at the same time requires computationally complex decryption before data can be used in processing, which may be prohibitive in large datasets.

Using tokenization, when data are requested to be erased, removing the mapping information from the tokenization system (*i.e.*, data allowing to establish the connection between the personal identifying data and other types of personal data) causes the data to be effectively erased. In comparison to the cryptographic approach, which requires complex decryption before data can be used, tokenization does not increase the computational complexity of data processing. However, since tokenization leaves the data accessible after erasure of mapping links, it thus requires careful design of the data structures in order to eliminate all connections between the personal and identifying information.

### Incentivisation

Positive and negative incentivization mechanisms can be used to promote eventual (long-term) unavailability of specified content. A soft negative incentivization, which does not require any changes to the underlying file sharing network, works as follows. When an erasure is requested, it is stored as a transaction on the public chain, thereby accessible by all storage nodes. At that moment, sharing of the erased content is forbidden: any further sharing of the content can be included as part of an “evidence transaction”, recorded on the main chain [8]. A successful evidence transaction leads to the stake of the perpetrator to be slashed. If the penalties are higher than the incentives collected for sharing data, storage nodes will not be motivated to serve “erased” data. Consequently, storage nodes can only store private copies or disseminate it out-of-band. However, if the primary purpose of the storage node is to dedicate its storage resources to the file sharing network, it will eventually evict the data from its system and store newer information that can be monetized. From a public standpoint, the data is effectively forgotten from the system in a probabilistic manner, depending on various factors, such as the rate of incoming data, the average amount of storage resources per node, the average stake deposited per node, and the penalty for serving erased data.

A positive incentivization mechanism requires modifications to off-chain file sharing networks. We demonstrate the modifications using Swarm. An erasure transaction on the main chain will record a bogus hash value associated to the content address. When a Swarm storage node is serving a piece of data, the content hash is verified against the main ledger. If the hashes do not match, the reward for serving content cannot be collected. Thus, storage nodes are incentivized not to store erased data, since they can no longer collect the reward associated with serving them. Furthermore, main chain peers will eventually prune the original transaction (containing the content address) from the Merkle tree of that block, since that address is no longer available or monetizable from the underlying file sharing network. New nodes doing a full sync on the blockchain will simply receive a pruned version of the blockchain without the erased data. Old nodes can still retain the erased content addresses, but will eventually forget them during garbage collection processes.

The incentivization mechanisms provide cryptoeconomic incentives to forgetting erased data. However, irrational agents may still choose to keep the data, so it can only be considered probabilistic. We believe such a mechanism satisfies the GDPR requirement of erasure without undue delay.

#### 4.7.6 Right to Data Access and Rectification

Currently, information about personally identified data needs to be requested from the company's individual employees and administrators, which is a very time-consuming (and therefore costly) and also a naturally error-prone approach. The use of a DBMS and its extended log recording on a blockchain would allow an easier response for a data access request. People who are interested in getting information about how their personal data is used within an organization under GDPR's right to data access could simply query the blockchain-based data storage, thus getting this information. The query could be both implemented off-chain as well as be provided as a smart contract. In the latter case, transparency would be given and the user does not have to implement a querying mechanism by herself. However, this approach requires all personal data to be traceable, *e.g.*, by using the same personal ID for all data related to a particular person.

The positive incentivization mechanism can also be used for the right of rectification, by replacing an existing piece of data with a newer one, which will now be monetized.

#### 4.7.7 Conclusion

This work provides a technical support to an effective implementation of the GDPR regulation, using techniques that are based on both blockchain and databases technologies. The paper presents GDPR, the relevant technologies, and the way we envision these technologies can serve an organization in becoming GDPR-compliant.

In our proposed solution we took advantage of the main benefits of each of the two named technologies. From databases, we take their ability to maintain data consistency, as well as its ability to efficiently store and retrieve large amounts of data. Blockchain's ability to secure transactions and keep a trustable ledger complement the set of needed abilities.

There are many open questions left to be discussed in future research. Especially, when realizing the proposed technology many issues may rise. For example, how to establish smart contracts between partners of different roles in exchanging data under GDPR? Also, how to realize (distributed) logging between a) controllers and processors, b) processors and processors, c) joint controllers based on blockchains?

## References

- 1 Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the Protection of Natural Persons with Regard to the Processing of Personal Data and on the Free Movement of Such Data, and Repealing Directive 95/46/EC (General Data Protection Regulation). *Official Journal of the European Union*, L119:1–88, April 2016.
- 2 Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. Hyperledger fabric: A distributed operating system for permissioned blockchains. *arXiv preprint arXiv:1801.10228*, 2018.
- 3 Luigi Atzori, Antonio Iera, and Giacomo Morabito. The Internet of Things: A survey. *Computer Networks*, 54:2787–2805, 2010.
- 4 Arshdeep Bahga and Vijay Madiseti. *Blockchain Applications: A Hands-On Approach*. VPT, 2017.
- 5 Thomas Baier, Claudio Di Ciccio, Jan Mendling, and Mathias Weske. Matching events and activities by integrating behavioral aspects and label analysis. *Software and System Modeling*, 17(2):573–598, 2018.
- 6 Thomas Baier, Jan Mendling, and Mathias Weske. Bridging abstraction layers in process mining. *Information Systems*, 46:123–139, 2014.
- 7 Juan Benet. Ipfs-content addressed, versioned, p2p file system, 2014.
- 8 Bitfury. Proof of stake vs proof of work.
- 9 Ethereum Blog. Merkle in Ethereum. <https://blog.ethereum.org/2015/11/15/merkle-in-ethereum/>. Accessed: 2018-08-17.
- 10 Eric A. Brewer. A certain freedom: thoughts on the CAP theorem. In Andréa W. Richa and Rachid Guerraoui, editors, *PODC*, page 335. ACM, 2010.
- 11 Vitalik Buterin. Ethereum: A next-generation smart contract and decentralized application platform. URL <https://github.com/ethereum/wiki/wiki/%5BEnglish%5D-White-Paper>, 2014.
- 12 Christian Cachin and Marko Vukolić. Blockchains consensus protocols in the wild. *arXiv preprint arXiv:1707.01873*, 2017.
- 13 Jan Camenisch and Anja Lehmann. Privacy for distributed databases via (un)linkable pseudonyms. *IACR Cryptology ePrint Archive*, 2017:22, 2017.
- 14 Min Chen, Shiwen Mao, and Yunhao Liu. Big Data: A Survey. *Mobile Networks and Applications*, 19(2):171–209, 2014.
- 15 Konstantinos Christidis and Michael Devetsikiotis. Blockchains and smart contracts for the internet of things. *IEEE Access*, 4:2292–2303, 2016.
- 16 Cisco. The internet of things reference model, 2014. White Paper Draft.
- 17 Michael Coblenz. Obsidian: A safer blockchain programming language. In *Proceedings of the 39th International Conference on Software Engineering Companion (ICSE-C)*, pages 97–99. IEEE Press, 2017.
- 18 E. Damaggio, R. Hull, and R. Vaculín. On the equivalence of incremental and fixpoint semantics for business artifacts with guard-stage-milestone lifecycles. *Information Systems*, 38:561–584, 2013.
- 19 Ali Dorri, Marco Steger, Salil S. Kanhere, and Raja Jurdak. Blockchain: A distributed solution to automotive security and privacy. *IEEE Communications Magazine*, 55(12):119–125, 2017.
- 20 Marlon Dumas, Marcello La Rosa, Jan Mendling, and Hajo A. Reijers. *Fundamentals of Business Process Management, Second Edition*. Springer, 2018.
- 21 Gilbert Fridgen, Florian Guggenmoos, Jannik Lockl, Alexander Rieger, and André Schweizer. Developing an evaluation framework for blockchain in the public sector: The ex-

- ample of the german asylum process. In *Proceedings of 1st ERCIM Blockchain Workshop 2018*. European Society for Socially Embedded Technologies (EUSSET), 2018.
- 22 Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.
  - 23 Luciano García-Bañuelos, Alexander Ponomarev, Marlon Dumas, and Ingo Weber. Optimized execution of business processes on blockchain. In *BPM'17: International Conference on Business Process Management*, Barcelona, Spain, September 2017.
  - 24 Nitin Gaur, Luc Desrosiers, Petr Novotny, V Ramakrishna, Anthony O'Dowd, and Salman Baset. *Hands-On Blockchain with Hyperledger: Building decentralized applications with Hyperledger Fabric and Composer*. Packt Publishing Limited, 06 2018.
  - 25 Jones Granatyr, Vanderson Botelho, Otto Robert Lessing, Edson Emílio Scalabrin, Jean-Paul Barthès, and Fabrício Enembreck. Trust and reputation models for multiagent systems. *ACM Computing Surveys (CSUR)*, 48(2):27, 2015.
  - 26 Gideon Greenspan. Avoiding the pointless blockchain project. *Online at <https://www.multichain.com/blog/2015/11/avoiding-pointless-blockchain-project>*, 2015.
  - 27 Christoph Hochreiner, Stefan Schulte, Schahram Dustdar, and Freddy Lécué. Elastic Stream Processing for Distributed Environments. *IEEE Internet Computing*, 19(6):54–59, 2015.
  - 28 Richard Hull, Elio Damaggio, Fabiana Fournier, Manmohan Gupta, Fenno Terry Heath, Stacy Hobson, Mark Linehan, Sridhar Maradugu, Anil Nigam, Piyawadee Sukaviriya, et al. Introducing the guard-stage-milestone approach for specifying business entity lifecycles. In *International Workshop on Web Services and Formal Methods*, pages 1–24. Springer, 2010.
  - 29 Xiaolong Jin, Benjamin W. Wah, Xueqi Cheng, and Yuanzhou Wang. Significance and challenges of big data research. *Big Data Research*, 2:59–64, 2015.
  - 30 Audun Jøsang, Roslan Ismail, and Colin Boyd. A survey of trust and reputation systems for online service provision. *Decision support systems*, 43(2):618–644, 2007.
  - 31 Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Annual International Cryptology Conference*, pages 357–388. Springer, 2017.
  - 32 S. Kiyomoto, M. S. Rahman, and A. Basu. On blockchain-based anonymized dataset distribution platform. In *2017 IEEE 15th International Conference on Software Engineering Research, Management and Applications (SERA)*, pages 85–92, June 2017.
  - 33 Why You Shouldn't Use Airbnb: 8 Troubling Issues You Didn't Know. <https://www.theinvisibletourist.com/why-you-shouldnt-use-airbnb-issues-you-didnt-know/>. [Online resources].
  - 34 Kari Korpela, Jukka Hallikas, and Tomi Dahlberg. Digital supply chain transformation toward blockchain integration. In *50th Hawaii International Conference on System Sciences*. AIS Electronic Library (AISeL), 2017.
  - 35 Shancang Li, Li Da Xu, and Shanshan Zhao. The internet of things: a survey. *Information Systems Frontiers*, 17(2):243–259, 2015.
  - 36 Orlenys López-Pintado, Luciano García-Bañuelos, Marlon Dumas, and Ingo Weber. Caterpillar: A blockchain-based business process management system. In *BPM'17: International Conference on Business Process Management, Demo track*, Barcelona, Spain, September 2017.
  - 37 Orlenys López-Pintado, Luciano García-Bañuelos, Marlon Dumas, Ingo Weber, and Alexander Ponomarev. CATERPILLAR: A business process execution engine on the ethereum blockchain. Technical report, Institute of Computer Science, University of Tartu, 2018.
  - 38 Michael Luca. Designing online marketplaces: Trust and reputation mechanisms. *Innovation Policy and the Economy*, 17(1):77–93, 2017.

- 39 Juri Mattila, Timo Seppälä, Catarina Naucler, Riitta Stahl, Marianne Tikkanen, Alexandra Bådenlid, Jane Seppälä, et al. Industrial blockchain platforms: An exercise in use case development in the energy industry. Technical Report 43, The Research Institute of the Finnish Economy, 2016.
- 40 Jan Mendling, Ingo Weber, Wil M. P. van der Aalst, et al. Blockchains for business process management - challenges and opportunities. *ACM Transactions on Management Information Systems*, 9(1):4:1–4:16, 2018.
- 41 Sebastien Meunier. When Do You Need Blockchain? Decision Models. <https://medium.com/@sbmeunier/when-do-you-need-blockchain-decision-models-a5c40e7c9ba1>, 2016.
- 42 Trade Ministry of Economy and Industry. Evaluation Forms for Blockchain- Based System. [http://www.meti.go.jp/english/press/2017/pdf/0329\\_004a.pdf](http://www.meti.go.jp/english/press/2017/pdf/0329_004a.pdf), 2017.
- 43 Miranda Mourby, Elaine Mackey, Mark Elliot, Heather Gowans, Susan E. Wallace, Jessica Bell, Hannah Smith, Stergios Aidinlis, and Jane Kaye. Are ‘pseudonymised’ data always personal data? implications of the gdpr for administrative data research in the uk. *Computer Law & Security Review*, 34(2):22–33, April 2018.
- 44 Catherine Mulligan, Jennifer Zhu Scott, Sheila Warren, and J.P. Rangaswami. Blockchain beyond the hype: A practical framework for business leaders. [http://www3.weforum.org/docs/48423\\_Whether\\_Blockchain\\_WP.pdf](http://www3.weforum.org/docs/48423_Whether_Blockchain_WP.pdf), 2018.
- 45 Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.
- 46 American Institute of Certified Public Accountants. *Guide to Audit Data Analytics*. AICPA, 2017.
- 47 Adam J. Oliner, Archana Ganapathi, and Wei Xu. Advances and challenges in log analysis. *Commun. ACM*, 55(2):55–61, February 2012.
- 48 There’s No Middleman On This Blockchain-Based Version Of Airbnb. <https://www.fastcompany.com/40524021/on-this-blockchain-based-version-of-airbnb-theres-no-middleman>. [Online resources].
- 49 P4Titan. Slimcoin a peer-to-peer crypto-currency with proof-of-burn, 2014.
- 50 Morgen E Peck. Blockchain world-do you need a blockchain? this chart will tell you if the technology can solve your problem. *IEEE Spectrum*, 54(10):38–60, 2017.
- 51 Raghu Ramakrishnan and Johannes Gehrke. *Database Management Systems*. McGraw-Hill, Inc., New York, NY, USA, 3 edition, 2003.
- 52 Paul Resnick, Ko Kuwabara, Richard Zeckhauser, and Eric Friedman. Reputation systems. *Communications of the ACM*, 43(12):45–48, 2000.
- 53 Ana Reyna, Christian Martín, Jaime Chen, Enrique Soler, and Manuel Díaz. On blockchain and its integration with IoT. Challenges and opportunities. *Future Generation Computer Systems*, 88:173–190, 2018.
- 54 Kurt Sandkuhl and Janis Stirna. *Capability Management in Digital Enterprises*. Springer Cham, 2018.
- 55 Reza Shokri, Carmela Troncoso, Claudia Díaz, Julien Freudiger, and Jean-Pierre Hubaux. Unraveling an old cloak: k-anonymity for location privacy. In *WPES*, pages 115–118, 2010.
- 56 Yutian Sun, Wei Xu, and Jianwen Su. Declarative choreographies for artifacts. In *International Conference on Service-Oriented Computing (ICSOS)*, pages 420–434. Springer, 2012.
- 57 Tim Swanson. Consensus-as-a-service: a brief report on the emergence of permissioned, distributed ledger systems, 2015.
- 58 An Binh Tran, Qinghua Lu, and Ingo Weber. Lorikeet: A model-driven engineering tool for blockchain-based business process execution and asset management. In *BPM’18: International Conference on Business Process Management, Demo track*, Sydney, NSW, Australia, September 2018.
- 59 UltraNote. Absolute privacy at your fingertips.

- 60 Wil M. P. van der Aalst. *Process Mining - Data Science in Action, Second Edition*. Springer, 2016.
- 61 Wil M. P. van der Aalst, Guangming Li, and Marco Montali. Object-centric behavioral constraints. Technical report, Eindhoven University of Technology, 2017.
- 62 Wil M. P. van der Aalst, Mathias Weske, and Dolf Grünbauer. Case handling: a new paradigm for business process support. *Data & Knowledge Engineering*, 53(2):129–162, 2005.
- 63 Marko Vukolić. The quest for scalable blockchain fabric: Proof-of-work vs. bft replication. In *International Workshop on Open Problems in Network Security*, pages 112–125. Springer, 2015.
- 64 Marko Vukolic. Eventually returning to strong consistency. *IEEE Data Eng. Bull.*, 39(1):39–44, 2016.
- 65 Ingo Weber, Xiwei Xu, Régis Riveret, Guido Governatori, Alexander Ponomarev, and Jan Mendling. Untrusted business process monitoring and execution using blockchain. In *BPM'16: International Conference on Business Process Management*, Rio de Janeiro, Brazil, September 2016.
- 66 Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger eip-150 revision (759dccc - 2017-08-07), 2017. Accessed: 2018-01-03.
- 67 Karl Wüst and Arthur Gervais. Do you need a blockchain? *IACR Cryptology ePrint Archive*, 2017:375, 2017.
- 68 Xiwei Xu, Ingo Weber, Mark Staples, Liming Zhu, Jan Bosch, Len Bass, Cesare Pautasso, and Paul Rimba. A taxonomy of blockchain-based systems for architecture design. In *Software Architecture (ICSA), 2017 IEEE International Conference on*, pages 243–252. IEEE, 2017.
- 69 Ennan Zhai, David Isaac Wolinsky, Ruichuan Chen, Ewa Syta, Chao Teng, and Bryan Ford. Anonrep: Towards tracking-resistant anonymous reputation. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, pages 583–596, Santa Clara, CA, 2016. USENIX Association.
- 70 Kaiwen Zhang and Hans-Arno Jacobsen. Towards dependable, scalable, and pervasive distributed ledgers with blockchains. In *38th IEEE International Conference on Distributed Computing Systems*, pages 1337–1346. IEEE Computer Society, 2018.
- 71 Kaiwen Zhang, Roman Vitenberg, and Hans-Arno Jacobsen. Deconstructing blockchains: Concepts, systems, and insights. In *Proceedings of the 12th ACM International Conference on Distributed and Event-based Systems*, pages 187–190. ACM, 2018.

## Participants

- Michael Coblenz  
Carnegie Mellon University –  
Pittsburgh, US
- Søren Debois  
IT University of  
Copenhagen, DK
- Claudio Di Ciccio  
Wirtschaftsuniversität Wien, AT
- Alevtina Dubovitskaya  
EPFL – Lausanne, CH
- Marlon Dumas  
University of Tartu, EE
- Fabiana Fournier  
IBM – Haifa, IL
- Avigdor Gal  
Technion – Haifa, IL
- Luciano García-Bañuelos  
University of Tartu, EE
- Stephan Haarmann  
Hasso-Plattner-Institut –  
Potsdam, DE
- Richard Hull  
IBM TJ Watson Research Center  
– Yorktown Heights, US
- Hans-Arno Jacobsen  
TU München, DE
- Mieke Jans  
Hasselt University, BE
- Agnes Koschmider  
KIT – Karlsruher Institut für  
Technologie, DE
- Qinghua Lu  
Data61, CSIRO – Sydney, AU
- Raimundas Matulevičius  
University of Tartu, EE
- Jan Mendling  
Wirtschaftsuniversität Wien, AT
- Petr Novotny  
IBM TJ Watson Research Center  
– Yorktown Heights, US
- Sooyong Park  
Sogang University – Seoul, KR
- Stefanie Rinderle-Ma  
Universität Wien, AT
- Stefan Schulte  
TU Wien, AT
- Jerome Simeon  
Clause Inc. – New York, US
- Ludwig Stage  
Tübingen, DE
- Mark Staples  
Data61, CSIRO – Eveleigh, AU
- Barbara Weber  
Technical University of Denmark  
– Lyngby, DK
- Ingo Weber  
Data61, CSIRO – Sydney, AU
- Francesca Zerbato  
University of Verona, IT
- Kaiwen Zhang  
ETS – Montreal, CA

