Introducing Trust in Service-oriented Distributed Systems through Blockchain

Marco Autili, Francesco Gallo, Paola Inverardi, Claudio Pompilio, Massimo Tivoli
Department of Information Engineering, Computer Science and Mathematics
University of L'Aquila
L'Aquila, Italy
{marco.autili,francesco.gallo,paola.inverardi,claudio.pompilio,massimo.tivoli}@univaq.it

***Abstract*—Business process management is concerned with the design execution, improvement, and monitoring of business processes. Systems that support the enactment and execution of processes have extensively been used by companies to streamline and automate *intra*-organizational processes. However, today's business enterprises must deal with global competition, heterogeneity, and rapidly develop new services and products. To address these requirements, the services reuse-based approach allowed enterprises to reconsider and optimize the way they do business, and change their information systems and applications to support collaborative business processes. Service choreographies support the reuse-based service-oriented philosophy in that they represent a powerful and flexible approach to realize systems by (possibly) reusing services and composing them in a fully distributed way. Nevertheless, for *inter*-organizational processes, challenges of collaborative design and lack of mutual trust have hampered a broader uptake. In this paper, we show an early stage approach to address the problem of trust in services choreography by using Blockchain technologies, in order to support the decentralized and peer-to-peer collaboration in a trustworthy manner, even in a network without any mutual trust between nodes.**

***Index Terms*—Blockchain, Distributed Systems, Services Oriented Computing, Trust**

I. Introduction

Collaboration has become critical to business success because of three major aspects: *segmentation of customers,* many customers want customized solutions requiring significant integration of knowledge; *information,* the rate and volume of information that people need to synthesize and apply has increased dramatically; *competition in the marketplace,* the global economy has increased competition [1]. By definition collaboration is "working with someone to produce or create something". In other words, it involves the sharing of resources, information, risks and responsibilities. Collaboration requires participants' mutual engagement and trust to achieve a common objective.

In the context of Information Technology (IT), these aspects are tackled by the business process management. It is a framework for streamlining business in organisation and inter-organisation, to improve the overall transparency and efficiency. In business process management, distributed computing is becoming more and more prevalent [2]. Moreover, as a result of the constant progress of the Internet, the systems are becoming more and more developed by integrating existing software, and reuse-based software engineering is becoming the primary development approach for building business and commercial systems [3], [4]. Services play a central role in this vision as effective means to achieve interoperability among parties of a business process, and new systems can be built by reusing and composing existing services and things. Service choreographies support the reuse-based service-oriented philosophy in that they represent a powerful and flexible approach to realize systems by (possibly) reusing services, and composing them in a fully distributed way.

During the last decade, with the aim of bringing the adoption of choreographies to the development practices adopted by IT companies, we focused our research and development activity on practical and automatic approaches to support the realisation of reuse-based service choreographies [3], [5]–[12]. In particular, a lot of attention was dedicated to the automatic synthesis of coordination logic and adaptation between participants involved in a business process described by a choreography. Although choreographies, by their very nature, guarantee decentralized collaboration, and then neutrality, they are not able to address the following question: (**Q1**) *how can it be ensured that one or more parties involved in the collaboration do not behave fraudulently, or that data created and modified in information systems outside the organisation are not tampered with, without delegating control to third parties or specific parties involved in the process itself*?

A. Novel Contribution

Q1 is a significant aspect in the context of distributed systems relying on the software reuse. In fact, often, the reused software is black box and it can be challenging to detect any fraudulent behavior or verify the identity of the participant that is providing a particular service. Traditionally, stand-alone computers and small networks rely on user authentication and access control to provide security. These physical methods use system-based controls to verify the identity of a person or process, explicitly enabling or restricting the ability to use, change, or view a computer resource. However, these strategies are inadequate for the increased flexibility that distributed networks, such as the Internet and pervasive computing environments, require because they lack of central control and their users are not all predetermined [13].

978-1-7281-5138-0/19/$31.00 ©2019 IEEE
DOI 10.1109/ISSREW.2019.00065

The integration between service-oriented context and blockchain technologies presents promising prospects, but it is still in an initial and challenging phase [14]. To this extent, the blockchain can be exploited as a new type of distributed computing platform by putting into communication smart contracts. In this paper, we address the problem of trust in services choreography using blockchain, Smart Contract and Validation Oracles technologies. We present an early stage approach to integrate the concept of Coordination Delegates (**CDs**) (Section III) within the "authorized" blockchain, in order to ensure trust within service choreographies without the help of a centralized authority or third parties. Our solution, influenced by [15], allows to coordinate and verify the correctness of the messages exchanged between the participants and to trust the involved participants. To motivate our approach, we describe a use case inspired by [16] for product traceability in supply chain networks. In supply chain management, the usage of blockchain-based application enhances tracking mechanisms and traceability assurance [17]. The use case concerns the tracking of products across supply chains to automate regulatory-compliance checking.

B. Paper Structure

The paper is structured as follow. Section II overviews blockchain technologies and choreographies. Section III describes the use of blockchain technology to realize trusted choreography–based systems. Section IV describes a use case for product traceability in supply chain networks and the related choreography-based system obtained by applying our approach. Conclusions and Future Works are given in Section V.

II. Background

This section presents general background information about choreographies, blockchain, smart contracts and blockchain oracle technologies.

A. Services Choreography

The need for choreographies was recognized in the Business Process Modeling Notation 2.0[1] (BPMN2), which introduced *Choreography Diagrams* to offer choreography modeling constructs. Choreography diagrams specify the message exchanges among the choreography participants from a global point of view. In fact, a choreography diagram models a distributed process specifying activity flows where each activity represents a message exchange between two participants. Graphically, a choreography task is denoted by a rounded-corner box with two bands representing the participant involved in the interaction captured by the task. The white band is used for the participant initiating the task that sends the initiating message to the receiving participant in the dark band, that can optionally send back the return message (e.g. considering the choreography on the left-hand side of Figure 1, the `T1` task where `A` sends the `M1` message to `B` that then send back the return message `M2`). A participant role models the expected interaction protocol that a concrete service should support in order to play it in the choreography. Depending on roles, BPMN2 choreography diagrams permit to distinguish among client participants (consumers), server participants (providers), and client-and-server participants (prosumers).

A choreography diagram models peer-to-peer communication by defining a multi-party and multi-role message-passing protocol that, when put in place by the concrete services playing the participant roles, permits to reach the choreography goal in a distributed way. It follows that, when third-party participants are involved, mostly back-box services being reused, obtaining the coordination logic required to realize a reuse-based choreography is a non-trivial and error-prone task.

B. Blockchain

A blockchain is a distributed ledger of transactions implemented as data batched into blocks that use cryptographic validation to link the blocks together. Each block references and identifies the previous block using a hashing function which forms an unbroken chain (i.e., blockchain).

In general, a public blockchain is neither stored in one central computer nor managed by any central entity. Instead, it is distributed and maintained by multiple computers or nodes (miners) that compete to validate the newest block entries and gain a reward [18]. The block validation system is designed to be immutable by a consensus protocol [19] that enables the nodes of the blockchain network to create trust in the state of the log and makes blockchains inherently resistant to tampering: all transactions are preserved forever without the possibility of being deleted. Anyone on the network can browse via a designated website and see the ledger. This provides a way for all participants to have an up-to-date ledger that reflects the most recent transactions or changes. In this way, blockchain establishes trust that facilitates transactions and brings many cost-saving efficiencies to all types of transactional interactions. The blockchain is pseudo-anonymous [20] because the identity of those involved in the transaction is represented by an address key in the form of a random string.

The first generation of blockchain provided a public ledger to store cryptographically-signed financial transaction [21]. There was very limited capability to support programmable transactions, and only very small pieces of auxiliary data could be embedded in the transactions to serve other purposes, such as representing digital or physical assets. The second generation of blockchain provides a general purpose programmable infrastructure with a public ledger that records the computational results. Programs that can be deployed and run on a blockchain are known as *smart contracts* (see Section II-C).

There are three types of blockchain: *public*, *private*, and *authorized*. These three types of blockchain can be combined in different ways, to best meet the needs of the user. The following are the most common combinations: **public permissionless blockchain (without authorizations)**, whose most

[1] http://www.omg.org/spec/BPMN/2.0.2/

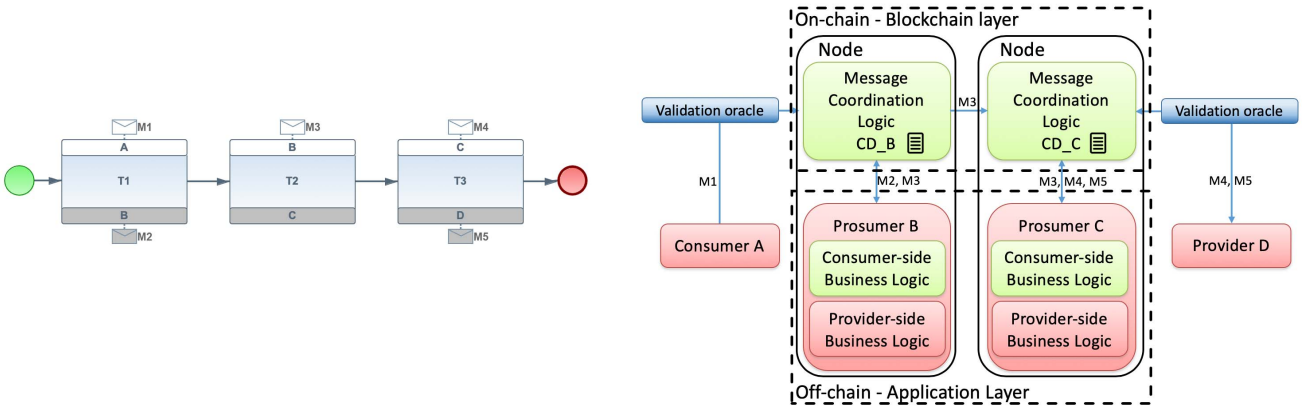

Fig. 1: Trusted choreography-based system example

famous examples are Bitcoin[2] and Ethereum[3]. This type of network allows access to any user who decides to connect and participate, generating new transactions, performing the task of miner or simply reading the log of transactions stored. In this system the miners are anonymous, so they are not considered trusted partecipants. **Public permissioned blockchain (authorized)**, such as Ripple[4]. It is a network that operates on behalf of a community that shares a common interest, where access to the role of miner is limited to a small number of individuals considered trusted. The level of log reading and participation in generating new transactions may or may not be limited, depending on the organization that controls the Blockchain. **Private permissioned blockchain (authorized)**, such as Chain[5]. In this case, only authorized and authenticated users can access the network because the blockchain operates exclusively within the limits of a well-defined community where all participants are known. Usually, these systems are leaded by financial institutions or government agencies that define who can access the network or not: this means that all miners are considered trustworthy.

C. Smart contract

In a traditional sense, a contract is an agreement between two or more parties in the form of a set of rules that the parties must obey. Thus, each party must trust the other party to fulfil its side of the obligation. Smart contracts [22], [23] provide the same kind of agreement but they remove the need for trust between parties. This is because a smart contract is codified and the results of its execution can be automatically verified. In fact, the three main elements of smart contracts are *autonomy*, *self-sufficiency*, and *decentralization*. Autonomy means that after it is launched and running, a contract and its initiating agent need not be in further contact. Smart contracts might be self-sufficient in their ability to marshal resources, that is, raising funds by providing services or issuing equity, and spending them on needed resources, such as processing power or storage. Smart contracts are decentralized in that they do not depend on a single centralized server; they are distributed and self-executing across network nodes. Smart contracts [24] are an important part of a blockchain framework and they make transactions traceable, transparent, and irreversible. Furthermore, smart contracts can express triggers, conditions and business logic [15] to enable complex programmable transaction.

D. Blockchain Oracles

Blockchains and smart contracts cannot access data from outside of their network. Indeed, smart contracts often need to access information from the outside world that is relevant to the contractual agreement, in the form of electronic data, by means of oracles [25], [26]. These oracles are services that send and verify real world occurrences and submit this information to smart contracts, triggering state changes on the blockchain.

Oracles feed the smart contract with external information that can trigger predefined actions of the smart contract. This external data stems either from software or hardware. Such a condition could be any data, like weather temperature, successful payment, or price fluctuations. However, it is important to note that a smart contract does not wait for the data from an outside source to be feed into the system. The contract has to be invoked, which means that one has to spend network resources for calling data from the outside world.

Oracles are also capable of digitally signing the data proving that the source of the data is authentic. Smart contracts can then subscribe to the oracles, and the smart contracts can either pull the data or oracles can push the data to the smart contracts. It is also necessary that oracles should not be able to manipulate the data they provide and must be able to provide authentic data.

There are different types of oracles, the main are: **Software Oracles**, handle information data that originates from online sources, like temperature, prices of commodities and goods,

[2]https://bitcoin.org/en/
[3]https://www.ethereum.org
[4]https://www.ripple.com
[5]https://chain.com

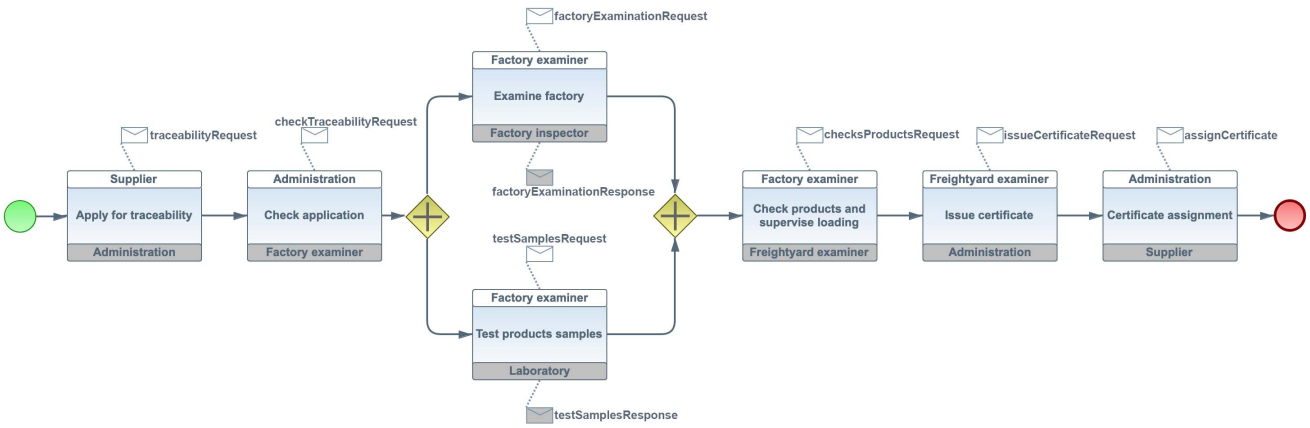

Fig. 2: Product Traceability choreography

train delays, etc. The software oracle extracts the needed information and pushes it into the smart contract. **Hardware Oracles**, some smart contracts need information directly from the physical world, for example, a car crossing a barrier where movement sensors must detect the vehicle and send the data to a smart contract, or RFID sensors in the supply chain industry. **Outbound Oracles**, provide smart contracts with the ability to send data to the outside world. An example would be a smart lock in the physical world, which receives a payment on its blockchain address and needs to unlock automatically.

Our approach is based on public blockchain since it is more suitable in stimulating business collaborations to improve business processes involving multiple parties, possibly reusing existing third-party services. Moreover, it is based on a permissioned blockchain (authorized) since it employs a more efficient validation process than the permissionless blockchain. In fact, the validation process in a permissioned blockchain is restricted only to certain nodes and hence this results in a high transaction throughput.

III. Approach

This section describes the exploitation of the blockchain technology within the approach defined in [12] to realize trusted choreography-based systems. The approach in [12] employs two types of connectors: communication channels and Coordination Delegates (CDs). The former are used to enable basic types of interaction among the services involved in the choreography; the latter enable complex (and distributed) coordination among services. The composition of CDs and their related communication channels realizes, in a distributed way, the glue code [27] that allows the involved services to interact with each other according to the specified choreography. The connectors in [12] provide communication and coordination services. In line with [28], we consider the blockchain as a network-based software connector which provides, beyond communication and coordination services, facilitation services. The facilitation services considered in our approach are: transaction validation and permission management. A transaction in a blockchain represents a message exchange and its integrity is guaranteed by cryptographic techniques. In particular, in case a transaction involves nodes of the blockchain, it is signed by the sender node with its signature and then the receiver validates the transaction or discards it if invalid. In case the transaction involves an external service, the validation activity is realized by interacting with a validation oracle. In particular, in case of an inbound transaction, if it involves a software service the validation oracle is a software oracle; instead, if the transaction involves a sensor the validation oracle is a hardware oracle. Otherwise, in case of an outbound transaction the validation oracle is an outbound oracle. These types of validation oracle can be combined, e.g. an inbound/outbound oracle that validates the related transactions. A valid transaction inside the blockchain triggers the execution of a smart contract, that is correctly coordinated by interacting with the ledger to verify the execution state of the choreography-based system. Finally, the transaction is propagated to other nodes in order to be recorded on the blockchain after reaching the consensus. As already introduced, our approach realizes choreography-based system as permissioned blockchains and hence the permission management allows to identify the nodes able to submit and validate transactions. Indeed, since we employ a public blockchain, unauthorized users could accidentally submit a transaction and hence a permission control mechanism is needed. In our approach, this mechanism is derived from the choreography specification and embedded into the validation oracles and the smart contracts running on the nodes of the blockchain. By introducing these facilitation services our approach realizes trusted choreography-based systems. Indeed, it addresses the question **Q1** defined in section I, in that it ensures the integrity and the trust of the transactions among the entities involved in the system.

Figure 1 shows the trusted choreography-based system, as obtained by applying our approach, corresponding to the choreography on the left. The system involves a consumer A, the prosumers B and C, and a provider D. The approach in [12] decouples the coordination logic of the CDs from the business

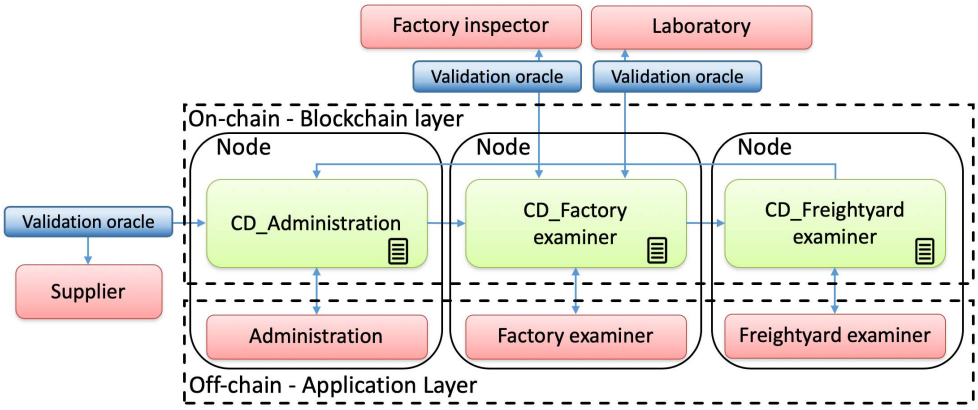

Fig. 3: Product Traceability choreography-based system

logic of the involved prosumer services. The latter is split into provider-side business logic, which can be implemented either from scratch or through reuse of existing services, and consumer-side business logic that is partially generated. As specified in [28], each node of the blockchain has two layers: blockchain layer implemented inside the blockchain (On-chain) and application layer implemented outside the blockchain (Off-chain). The blockchain layer contains CDs, whose coordination logic is implemented as smart contracts, while the application layer contains the prosumers business logic. It is worth noting that the figure shows the case in which the provider-side business logic of B and C is implemented from scratch and hence it is contained inside the nodes of the blockchain. In case the provider-side business logic is reused, it is implemented outside the blockchain.

Considering the choreography task `T1`, the message M1 sent by the consumer A is authorized and validated through the validation oracle and then it is received by CD_B that forwards it to the provider-side business logic. Then, in order to suitably coordinate the task `T2`, CD_B gets the message M3 by the consumer-side business logic and exchanges it with CD_C. After successfully authorizing and validating the transaction, CD_C forward M3 to the provider-side business logic. Finally, to realize the choreography task `T3`, CD_C obtains the message M4 from the consumer-side business logic and then sends it to the provider D. The latter replies with the message M5 that is then sent to the consumer-side business logic of the prosumer B. The transaction between CD_C and the provider D is authorized and validated by means of a validation oracle.

IV. Case Study

This section introduces a use case for product traceability in supply chain networks. It is inspired by [16] and it concerns the tracking of products across supply chains to automate regulatory-compliance checking.

Figure 2 shows the Product Traceability choreography specification by means of a BPMN2 Choreography Diagram. The use case starts with a product supplier applying for traceability services by interacting with the administration of the traceability service provider. Then, the administration asks a factory examiner for inspecting the products throughout the supply chain of the supplier. To this extent, the factory examiner performs two parallel verifications (see the parallel branch represented as a diamond marked with a "+", with two outgoing arrows after the choreography task `Check application`) by delegating to a factory inspector the examination of the factory and to a laboratory the testing of products samples.

The last verification involves the freight yard examiner that checks the products and supervises the loading before issuing a certificate to the administration of the traceability service provider. Finally, the administration assigns a certificate to the supplier stating that its products comply with the regulation with regard to their origin and quality.

Figure 3 shows the choreography-based system realizing the product traceability choreography. It consists of the CDs for the participants: `Administration`, `Factory examiner` and `Freightyard examiner`. Each of these CD is contained into a node of the blockchain together with the related prosumer business logic that is implemented from scratch. The services corresponding to the participants: `Supplier`, `Factory inspector` and `Laboratory` are third-party external services reused and hence they interact with the related CDs by means of an inbound/outbound validation oracle. In this choreography-based system, certified laboratories, inspectors, examiners, suppliers and traceability service providers collaborate in a trustworthy manner to realize a traceability system. The application layer contains private data and raw data, e.g. supplier information, raw files of traceability certificates and photos, whereas the blockchain layer contains traceability information required by the traceability regulation, e.g. traceability results and inspection date.

V. Conclusion and Future Works

The use of blockchains in the service-oriented context is still in its early stages and presents numerous challenges. In this paper, we presented an approach based on our previous

works, to address the problem of trust in services choreography using blockchain technologies. In order to ensure trust within service choreographies without the help of a centralized authority or third parties, our solution allows to coordinate and verify the correctness of the messages exchanged between the participants and to trust the involved participants.

To assess the feasibility and usability of our approach, we plan to identify technical integration points between it and the hyperledgers technology[6] to create a prototype implementation deployed on a blockchain network and to evaluate its performance against several case studies.

ACKNOWLEDGMENTS

This research work has been supported by (i) the Ministry of Economy and Finance, Cipe resolution n.135/2012 (INCIPICT - INnovating CIty Planning through Information and Communication Technologies), (ii) the GAUSS national PRIN project (Contract No. 2015KWREMX), and (iii) the Italian "POR-FSER Abruzzo 2014-2020" ConnectPA project.

REFERENCES

[1] M. M. Beyerlein and C. L. Harris, *Guiding the journey to collaborative work systems : a strategic design workbook*. San Francisco, Calif. : Pfeiffer, 2004, includes index.

[2] D. Georgakopoulos, M. Hornick, and A. Sheth, "An overview of workflow management: From process modeling to workflow automation infrastructure," *Distributed and Parallel Databases*, vol. 3, no. 2, pp. 119–153, Apr 1995. [Online]. Available: https://doi.org/10.1007/BF01277643

[3] M. Autili, P. Inverardi, and M. Tivoli, "Automated synthesis of service choreographies," *IEEE Software*, vol. 32, no. 1, pp. 50–57, 2015. [Online]. Available: https://doi.org/10.1109/MS.2014.131

[4] M. Güdemann, P. Poizat, G. Salaün, and L. Ye, "Verchor: A framework for the design and verification of choreographies," *IEEE Transactions on Services Computing*, vol. 9, no. 4, pp. 647–660, July 2016.

[5] J. Cámara, K. L. Bellman, J. O. Kephart, M. Autili, N. Bencomo, A. Diaconescu, H. Giese, S. Götz, P. Inverardi, S. Kounev, and M. Tivoli, *Self-aware Computing Systems: Related Concepts and Research Areas*. Cham: Springer International Publishing, 2017, pp. 17–49. [Online]. Available: https://doi.org/10.1007/978-3-319-47474-8_2

[6] M. Autili, P. Inverardi, R. Spalazzese, M. Tivoli, and F. Mignosi, "Automated synthesis of application-layer connectors from automata-based specifications," *Journal of Computer and System Sciences*, vol. 104, pp. 17 – 40, 2019, language and Automata Theory and Applications - LATA 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0022000019300248

[7] M. Autili, A. Di Salle, and M. Tivoli, "Synthesis of resilient choreographies," in *Software Engineering for Resilient Systems*, A. Gorbenko, A. Romanovsky, and V. Kharchenko, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 94–108.

[8] M. Autili, D. Di Ruscio, A. Di Salle, P. Inverardi, and M. Tivoli, "A model-based synthesis process for choreography realizability enforcement," in *Fundamental Approaches to Software Engineering*, V. Cortellessa and D. Varró, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 37–52.

[9] M. Autili, L. Mostarda, A. Navarra, and M. Tivoli, "Synthesis of decentralized and concurrent adaptors for correctly assembling distributed component-based systems," *Journal of Systems and Software*, vol. 81, no. 12, pp. 2210 – 2236, 2008, best papers from the 2007 Australian Software Engineering Conference (ASWEC 2007), Melbourne, Australia, April 10-13, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121208000770

[10] M. Autili, P. Inverardi, and M. Tivoli, "Choreography realizability enforcement through the automatic synthesis of distributed coordination delegates," *Science of Computer Programming*, vol. 160, pp. 3 – 29, 2018, fundamentals of Software Engineering (selected papers of FSEN 2015). [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167642317302228

[11] M. Autili, A. Di Salle, F. Gallo, C. Pompilio, and M. Tivoli, "Model-driven adaptation of service choreographies," in *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, ser. SAC '18. New York, NY, USA: ACM, 2018, pp. 1441–1450. [Online]. Available: http://doi.acm.org/10.1145/3167132.3167287

[12] ——, "Aiding the realization of service-oriented distributed systems," in *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, ser. SAC '19. New York, NY, USA: ACM, 2019, pp. 1701–1710. [Online]. Available: http://doi.acm.org/10.1145/3297280.3297446

[13] L. Kagal, T. Finin, and A. Joshi, "Trust-based security in pervasive computing environments," *Computer*, vol. 34, no. 12, pp. 154–157, Dec 2001.

[14] F. Daniel and L. Guida, "A service-oriented perspective on blockchain smart contracts," *IEEE Internet Computing*, vol. 23, no. 1, pp. 46–53, Jan 2019.

[15] I. Weber, X. Xu, R. Riveret, G. Governatori, A. Ponomarev, and J. Mendling, "Untrusted business process monitoring and execution using blockchain," in *Business Process Management*, M. La Rosa, P. Loos, and O. Pastor, Eds. Cham: Springer International Publishing, 2016, pp. 329–347.

[16] Q. Lu and X. Xu, "Adaptable blockchain-based systems: A case study for product traceability," *IEEE Software*, vol. 34, no. 6, pp. 21–27, November 2017.

[17] F. Casino, T. K. Dasaklis, and C. Patsakis, "A systematic literature review of blockchain-based applications: Current status, classification and open issues," *Telematics and Informatics*, vol. 36, pp. 55 – 81, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0736585318306324

[18] M. Sharples and J. Domingue, "The blockchain and kudos: A distributed system for educational record, reputation and reward," in *Adaptive and Adaptable Learning*, K. Verbert, M. Sharples, and T. Klobučar, Eds. Cham: Springer International Publishing, 2016, pp. 490–496.

[19] C. Cachin and M. Vukolic, "Blockchain consensus protocols in the wild," *CoRR*, vol. abs/1707.01873, 2017. [Online]. Available: http://arxiv.org/abs/1707.01873

[20] S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko, D. McCoy, G. M. Voelker, and S. Savage, "A fistful of bitcoins: Characterizing payments among men with no names," in *Proceedings of the 2013 Conference on Internet Measurement Conference*, ser. IMC '13. New York, NY, USA: ACM, 2013, pp. 127–140. [Online]. Available: http://doi.acm.org/10.1145/2504730.2504747

[21] M. Swan, *Blockchain: blueprint for a new economy*. Sebastopol, Calif.: O'Reilly Media, 2015. [Online]. Available: http://shop.oreilly.com/product/0636920037040.do

[22] N. Szabo. (1996) Smart contracts: Building blocks for digital markets. [Online]. Available: http://www.alamut.com/subj/economics/nick_szabo/smartContracts.html

[23] R. M. Parizi, Amritraj, and A. Dehghantanha, "Smart contract programming languages on blockchains: An empirical evaluation of usability and security," in *Blockchain – ICBC 2018*, S. Chen, H. Wang, and L.-J. Zhang, Eds. Cham: Springer International Publishing, 2018, pp. 75–91.

[24] ——, "Smart contract programming languages on blockchains: An empirical evaluation of usability and security," in *Blockchain – ICBC 2018*, S. Chen, H. Wang, and L.-J. Zhang, Eds. Cham: Springer International Publishing, 2018, pp. 75–91.

[25] "Chainlink: A decentralized oracle network," https://link.smartcontract.com/whitepaper, accessed: 2019-08-30.

[26] "Provable's oracle model," https://provable.xyz/, accessed: 2019-08-30.

[27] R. Allen and D. Garlan, "A formal basis for architectural connection," *ACM Trans. Softw. Eng. Methodol.*, vol. 6, no. 3, pp. 213–249, Jul. 1997. [Online]. Available: http://doi.acm.org/10.1145/258077.258078

[28] X. Xu, C. Pautasso, L. Zhu, V. Gramoli, A. Ponomarev, A. B. Tran, and S. Chen, "The blockchain as a software connector," in *2016 13th Working IEEE/IFIP Conference on Software Architecture (WICSA)*, April 2016, pp. 182–191.

[6] https://www.hyperledger.org