PenChain: A Blockchain-Based Platform for Penalty-Aware Service Provisioning

Trung-Viet Nguyen[a,c], Lam-Son Lê[a,b,*], Syed Attique Shah[d], Dirk Draheim[e]

[a] *Faculty of Computer Science and Engineering, Ho Chi Minh City University of Technology (HCMUT) 268 Ly Thuong Kiet Street, District 10, Ho Chi Minh City, Vietnam*
[b] *Vietnam National University (VNU-HCM), Linh Trung Ward, Thu Duc City, Ho Chi Minh City, Vietnam*
[c] *Faculty of IT, Can Tho University of Technology, Vietnam*
[d] *School of Computing and Digital Technology, Birmingham City University, Millennium Point, B4 7XG, Birmingham, United Kingdom*
[e] *Information Systems Group, Tallinn University of Technology, Akadeemia tee 15a 12618 Tallinn, Estonia*

Abstract

Service provisioning is of paramount importance in various business domains because we now head for a world of integrated services and the so-called service ecosystem. For instance, with the rise of the Internet of Things, agricultural practitioners, and data engineers expect to benefit from next-generation data provisioning and service engineering. A large number of service offerings that are made available to customers necessitate a novel approach to service registry and discovery that lets them choose the offerings that best match their preferences. One way to do this is to introduce the provider's reputation – a quality indicator of the provisioned service, as an additional search criterion. With blockchain technology now in our hands, computationally regulating service-level agreements (SLAs) that capture the mutual agreements between the customer has regained momentum. In this article, we report our full-fledged work on the conception, design, and construction of a platform for SLA-minded service provisioning called PenChain. We argue and demonstrate that the penalty-aware SLA of general services, if represented in a machine-readable logic and assisted by distributed ledger technology, is programmatically enforceable in PenChain. We report our further findings about the relevance, suitability, and scalability of PenChain.

Keywords: service provisioning; service-level agreements; smart contracts; precision agriculture; tourism; penalty

1. Introduction

Service provisioning is still on the rise in today's information systems as we head for a world of massively integrated services and the so-called service ecosystem. Service provisioning is of paramount importance in various business domains such as precision agriculture, supply chain management, and smart tourism [1]. For instance, with the rise of the Internet of Things (IoT), agricultural practitioners and data engineers expect to benefit from real-time data provisioning for the sake of advanced analytics. A customer who wishes to use a data service would choose between offerings that provide the same data packages (e.g., the pH of water, and humidity) but at different service levels. Similarly, a tourist might consider multiple tour offerings and

[*]Corresponding author
Email address: `lam-son.le@alumni.epfl.ch` (Lam-Son Lê)

picks the most favorable in terms of service levels. Once the service offerings that are made available in such a service ecosystem exceed a critical number, it needs a novel service registry and lookup mechanism[1].

Distinguishing functionally similar services using quality is vital for effectively browsing a large number of service offerings [2]. Once the customer has picked an offering, she literally enters a service contract with the provider. The service level at which this service contract comes into effect hence called the service-level agreement (SLA), captures the mutual agreements between the customer and the service provider. The recordings of these service levels over a period of time, if securely stored using a mutually trusted mechanism, would suggest how reputable a provider has been of late. To this end, search results viewed by the customer should be sorted by the reputation of the service providers before being further customized to match the customer's preference.

The concept of SLA has been investigated substantially in service-oriented cloud computing and telecommunications. Research on SLAs falls into the following directions [3, 4]: describing (i.e., formally representing the SLAs taking into account the penalty rules) such as iAgree[2], monitoring (keeping track of whether an SLA is respected during the execution of services), negotiation (mutually adapting an SLA while executing services to avoid a contract violation), and enforcement (executing the rule-based clauses stated in an SLA). Research on how to enforce the SLAs still stays on the sidelines compared to its counterparts primarily due to the lack of supporting technology. As expected, this research line has recently gained traction in the community of service engineering thanks to the emergence of distributed ledger technology.

Our research in this realm has resulted in preliminary work on a blockchain-based enforcing mechanism for penalty-aware SLAs that facilitates the execution of all penalty rules specified in an SLA. In case the provider fails to realize a penalty despite having tried all available alternatives, this enforcement mechanism records a breach of contract in an underlying ledger and accordingly updates the rule-abiding rate of the provider in question. Specifically, we proposed a mechanism for enforcing penalty-aware SLAs in the tourism sector [5] and a platform for digitizing the SLAs of data provisioning in IoT-enabled smart farming [6]. In this article, we report our full-fledged work on the conception, construction, and validation of such a platform for SLA-minded service lookup called PenChain.

We contribute:

(i) a formal definition for dynamic penalty-aware SLAs;

(ii) a ranking algorithm for sorting service offerings listed in a search result;

(iii) a distributed application architecture oriented towards Web3 that enforces the penalty-aware SLAs and objectively calculates the service providers' reputation[3].

Our further findings include the relevance and scalability of such an enforcement mechanism. We investigate the relevance of PenChain to several business domains. One of these domains is about the provisioning of IoT data services in smart shrimp farming that we elaborate in a case study. Overall, the enforcement mechanism of PenChain proves to be useful for handling the penalty-aware SLAs in at least two domains

[1]Think of such an ecosystem of data services as an e-commerce platform where shoppers search for shopping items by entering a couple of keywords and later on add them to their virtual shopping cart. The search results need to be presented in ways that maximize the purchase order placed by the shopper.

[2]iAgree Specification `http://iagree.specs.governify.io/`

[3]Many intermediary websites in use today still rely on customer's feedback to calculate the average rating of a service provider, possibly resulting in unfair assessment and biased reputation [7, 8]. Moreover, leaving feedback is a time-consuming exercise that often is ignored by the customers.

and scalable in real-life scenarios provided that the underlying private blockchain of PenChain comes with a matched capability.

We proceed as follows. Section 2 is dedicated to preliminaries and related work. In Section 3, we formulate our research statement. In Section 4, we propose our general framework for enforcing the penalty-aware SLAs and show that our SLA-enforcing proposal is useful by discussing its suitability for three example sectors. In Section 5, we offer an in-depth engineering analysis of PenChain for the provisioning of IoT data services. We analyze the scalability and suitability of our PenChain framework in the big picture of services computing in Section 6. In Section 7, we discuss the potential future directions of our study. Section 8 concludes the paper and outlines our future work.

2. State of the art

In this section, we summarize the scholarly work on SLA-enabled service management and its enabling technologies (blockchain, Web3).

2.1. Service-level agreements

SLA-related research has been studied extensively in fields such as engineering, economics, management science, and psychology. These articles can be categorized into the following four groups.

- *SLA definition.* Service-level agreements are a set of promises, guarantees, and obligations that the service provider makes to the client [9]. Service level agreements (SLAs) detail not just the service standards but also any agreed-upon remedies, penalties, or level requirements. Many attempts have been made to explain SLAs, but most of them have focused on online services and cloud computing. The online service-level agreement, also known as the WSLA, was first presented by IBM research as a framework for generating and monitoring SLAs [10]. Research on this SLA definition is more diversified in many different domains [11, 12, 13].

- *Real-time SLA monitoring.* The practice of ensuring that service level agreements are met is known as SLA monitoring. It might be performed by a third party or by the IT staff. Moreover, SLA monitoring is a radical resolution strategy in situations of SLA breaches and key commercial issues for the services industry [14]. For instance, Labidi [15] described a semantic SLA modeling and monitoring technique for cloud computing. Several theories have been proposed for the IoT [16, 17], some focusing on the blockchain [18], others on the 5G network [19].

- *SLA negotiation.* SLA negotiation is the process of negotiating the level of quality and service that is acceptable to both parties. It is often used in the context of information technology and software development. Some studies have focused on the subject of SLA negotiation, highlighting the responsibilities and problems involved, and proposing frameworks for resolving the issue with cutting-edge technology [20, 21, 22].

- *SLA enforcement.* Research in this area aims to enhance the management of SLAs by creating, monitoring, and encouraging users to report service faults. For example, the work conducted by Nakashima [23] leverages Ethereum to propose a collection of Web APIs that automate the SLA lifecycle enforcement on the blockchain. Furthermore, Zhou et al. [24] proposed a witness approach to the enforcement of SLAs using smart contracts. In the domain of IoT services, Alzubaidi [25] proposed a blockchain-based decentralized approach for assessing SLA compliance and enforcing consequences within cloud-based internet of things applications.

SLAM [26] is a self-contained and trustworthy framework for continuous SLA monitoring in a multi-cloud environment that is blockchain-based and employs smart contracts to discover SLA breaches as a result of these benefits. For this purpose, Abhishek et al. [27] presented a blockchain-based system that ensures the integrity of the client's logs and verifies SLA breaches, resulting in a reliable ecosystem. Neidhardt [28] introduced a blockchain-based monitoring mechanism to ensure customer trust in services. Blockchain technology would assist in monitoring the SLAs and improving service trust in cloud computing [29, 30, 31]. Similar solutions may be found in fog computing [32], edge computing [33], and 5G, 6G networks [34, 35]. Viewing the blockchain as a trusted ledger that records all service attributes, a blockchain-based service recommendation system [36] and an auditing solution to protect 5G consumer data [37] are prominent.

Smart contracts are increasingly employed to construct autonomous applications [38]. The public blockchain platform of Ethereum is recommended for SLA monitoring and penalty enforcement [39]. Such a framework has been proposed in [40] to monitor SLA terms and compensation in an automatic and decentralized manner using smart contracts and blockchain technologies. They have recommended that compensation should be set up either through basic notifications or automatically through a web application. Singh and Lee [41, 42] provided an approach for the blockchain cloud based on the SLA specification. It provided a more in-depth overview of the construction of smart contracts geared toward SLA. Uriarte et al. [43] presented a distributed SLA management using smart contracts and blockchain technology. This system features a distributed cloud offering dynamic services and promoting reduced costs for cloud consumers.

The aforementioned research has a goal in common: to build more open and egalitarian systems for customers and providers. While the studies we surveyed mostly examine SLAs in cloud computing SLAs over specific parameters, we investigate if a real-life SLA incorporates not only technically measurable indicators but also penalty rules that are contractually articulated to protect the service consumers' rights in many sectors. We follow the recently-emerging trend that we make the case for in this subsection. More specifically, we utilize distributed ledger technology in determining the reputation of the service providers and whether or not a provider adheres to the indemnification.

2.2. *Blockchain-enabled platforms for data collection and exchange*

IoT redefines existing protocols to establish a more connected network of devices, allowing data to be readily collected and exchanged even when there is no formal human-to-human or human-computer contact. That is why IoT data has a huge potential for software firms, and it will expand even more when blockchain technology is used. Liu et al. [44] asserted that blockchain could provide high-quality and secure data sharing for industrial IoT applications. In parallel, many studies are being conducted on blockchain platforms to address the data management and integrity concerns brought by IoTs in various industries [45, 46]. Besides, the blockchain was connected with edge computing servers to improve data quality and securely handle the compute-intensive activities demanded by IoT devices [47]. Ardagna et al. [48] presented a reliable data collection method based on blockchain technology and smart contracts. It attempts to filter out any data untrusted data trusted based on trust criteria. Moreover, this method would evaluate the state of IoT devices and the gathered data. The blockchain-based IoT solutions are ideal for streamlining company automation, achieving substantial cost savings, and improving the user experience [49]. Using the blockchain and SDN, Hameed et al. [50] proposed a scalable method for the IoT device key and trust management. Simulation demonstrates that this combination can store the public keys of IoT devices on the blockchain and efficiently route network traffic using SDN. Similarly, Hameed et al. [51] observed that with the assistance of AI, adaptive resource management frameworks for IoT networks would be developed, including blockchain-based SDN frameworks.

2.3. Web3

According to the current most standard explanation, the web has emerged in stages of Web 1.0 and Web 2.0, and is currently to be transformed into Web 3.0 [52]. In parallel, the vision of Web3 [53] is currently widely discussed by major technology analysts such as Gartner [53], Forrester [54] and Forbes Technology Council [55] as well as the Harvard Business Review [56, 57, 58]. Web3 needs to be carefully distinguished from the standard Web 3.0 vision[4] – although the objectives of Web3 and Web 3.0 share some commonalities.

According to the (rather minimal) standard explanation, the web started as a static web, named Web 1.0, which was about organizations sharing their content. Next, Web 2.0 was about enabling individuals to share their content. Practically, this can be connected to the emergence of social media platforms. Additionally, Web 3.0 [52], also coined the Semantic Web [59], is about turning the web into a web of knowledge. This means that knowledge presentation is standardized so that it becomes automatically processable. Consequentially, Web 3.0 is typically identified with the W3C (World Wide Web Consortium) standards RDF (Resource Description Framework) [5] and OWL (Web Ontology Language)[6]. Albeit this standard explanation reflects some of the most important aspects of the history of the web, it does not adequately grasp some other crucial, more complex evolvements. In particular, a major stage of the web, which is not reflected in the standard explanation, was the introduction of e-commerce. Technologically, this was enacted by the introduction of SSL (Secure Socket Layer), which was the necessary precondition to enable payments on the Internet [60]. E-Commerce was then also the driver of web technologies for data exchange and service computing [61]. Furthermore, Web 2.0 needs a closer look. The vision was about enabling individuals to create content on the web, however, originally in the vein of the peer-to-peer ("distributed") mentality of the early days of the web. What we have actually seen, however, is the emergence of tech giants (Big Tech) getting into control over the individuals' data – it was exactly this development that has led [62, 63] to propose the web decentralization project Solid[7] (Social Linked Data) [64, 65].

The Web3 vision takes blockchain disintermediation to a next level by making it ubiquitous, encompassing not only payments and financial services but also digital identities, data, and business models [66]. [57] have characterized the Web3 as "a decentralized, blockchain-based internet ecosystem owned and operated by its users" and their expectations are high towards Web3 being "our chance to make a better internet" [57]. We summarize a series of most significant Web3 characteristics as present in the current discourse by comparing each of them briefly with the current situation of the Web 2.0 [66, 67]:

- *Web3 Payments.* In the current web, payments are online bank transfers between accounts that are hosted by commercial banks. There exist "digital payments in existing currencies – through Paypal and other »e-money« providers such as Alipay in China, or M-Pesa in Kenya" [68], however, these digital payments still rely on bank transfers in the backend. In regards to today's tiered monetary system, payments are therefore done with M1-money – due to the necessary involvement of commercial banks [69]. Instead, in Web3, cryptocurrencies enable direct payments between web users, i.e., payments without intermediaries. The genuine Web3 currencies are neither owned by a central bank nor collateralized commercial bank money, i.e., they do not belong to the established monetary system and, therefore, cannot be classified as being M0- or M1-money. Note, that central bank digital currency [70] is usually not considered part of the Web3 vision.

[4]In some sources Web3 might be named Web 3.0; still, it then has to be distinguished from the standard Web 3.0 vision. But these sources get fewer and can be neglected. Henceforth, in this article, we use Web 3.0 for the standard Web 3.0 vision.

[5]https://www.w3.org/TR/rdf-syntax-grammar/

[6]https://www.w3.org/TR/owl-features/

[7]https://solidproject.org/

- *Financial Services.* Currently, financial services are not considered as being a part of the web, even though they are made accessible through web-based e-commerce services. Instead, the Web3 vision relies on built-in DeFi [71, 72, 73, 74]. Here, financial services are considered as an integral part of Web3 – disrupting both established commercial banking and investment banking.

- *Identity.* In the current web, online identities are created and linked to real-world identities via legally trusted entities, which rely on established routines of personal identity proofing [75]. These online identities rely on public key infrastructures (KPIs) or cloud-based identity solutions, each of them with a different level of technological and organizational maturity [76]. Authorities and companies serve as trust anchors in creating online identities. The Web3 stands for a paradigm shift and strives for self-sovereign identity [77] – consequentially extending the tradition of the peer-to-peer community [78]. Consensus protocols are seen to replace traditional trust anchors and, again, disintermediation is seen as the crucial notion.

- *Data Ownership.* In the current web, data is owned and utilized by companies. Instead, in the Web3 vision, data is owned and utilized by the users.

- *Business Models.* From the perspective of Web3, the current web is dominated by Silicon Valley tech giants such as Alphabet, Amazon, and Meta. Business models center around super-scaling e-commerce and social media/networks that commercialize the data of their customers. The Web3 envisions new business models [79] that are (i) based on new forms of organization such as the decentralized autonomous organization (DAO) [80] and/or (ii) rely on the utilization of genuine Web3 currencies (cryptocurrencies) or other Web3 assets such as NFTs (non-fungible tokens) [81, 82, 83]. Genuine DeFi business models (decentralized payment services, decentralized fundraising, decentralized contracting) are particularly important [73] for the Web3 vision.

The vision of a ubiquitous integration of emerging technology has become widely known as the Internet of Things (IoT) – the Web3 vision can be characterized as the *Web of Everything*, and even more, the *Web of Everything and Everybody*, since the idea of being “owned and operated by its users” [57] can be considered the key ingredient of Web3.

3. Research questions

In service computing, description languages for SLAs and has gained a lot of research attention. Due to a lack of technological support, executing an SLA that has been agreed upon by the provider and its consumers without human intervention remains one of the most challenging questions. Before we enter the digital transformation era, the mainstream thought on the subject was to introduce a regulatory entity whose main job is to monitor and referee service provisioning. This agent[8] keeps an eye on all service transactions and, in case of dispute, might invoke a pre-programmed unit to kick off a negotiation workflow to avoid a breach of contract. As distributed ledger technologies continue to advance at a rapid pace, enforcing the SLAs has now become a technologically backed research attempt. But before reaching this point, we will have to address a few research questions in the following. Let’s elaborate on these research questions in Subsections 3.1, 3.2, and 3.3.

[8]Centralized computing was particularly made popular in the Web2 era. In light of Web3, we should replace such an agent with a decentralized application that is meant to perform the same job.

- **RQ1**: Penalty rules specified in a dynamic SLA dictate what the provider and its customers should do in case of a dispute. To make these rules computer-interpretable, in what ground logic or formal languages shall we express them?
- **RQ2**: How to digitize the penalty-aware SLAs to enable an enforcement mechanism and the computation of an objective unbiased reputation in service provisioning?
- **RQ3**: How to gear up a distributed ledger for enforcing the relevant penalty rules during a service transaction?

3.1. RQ1: In what ground logic or representational technique would the penalty rules be expressed?

Handling penalty rules becomes increasingly important in service operations, as evidenced by a great amount of scholarly work in rule-based modeling and monitoring. We head for dynamic SLAs (which incorporate human-mediated factors such as the penalty) as opposed to static ones (which mainly describe uptime/downtime and availability constraints). While a penalty rule is in place to primarily protect the service customers, it should give the provider multiple chances to repair any spontaneous SLA violation during a service transaction. Should the provider run out of opportunities for taking action to fix such a violation, a breach of contract is finally recorded to report that the customer's expectation was not met. In other words, the penalty rule, being triggered by a spontaneous SLA violation, dictates a path for both the customer and the provider to follow through with the goal to fix the said violation and eventually discard it. We were in search of a ground logic or a formal language expressible enough to represent a penalty rule that articulates cascaded reparation actions.

3.2. RQ2: How to digitize the penalty-aware SLAs to enable an enforcement mechanism and the computation of an objective unbiased reputation in service provisioning?

The dependability of a service system or an ecosystem of services is in part linked to how we control the SLAs. We argue that, unlike many present-day's intermediary websites that rely on subjective customer feedback, the next-gen service ecosystem should employ a novel technique for objectively calculating the service providers' ratings using logs of computerized SLAs. Digitizing penalty-aware SLAs is non-trivial for the following reasons: (a) Suppose we have found an appropriate logic for the representation of the penalty rules, we still need a special computer program to fire the reparation actions articulated in these rules without human intervention; (b) Digital evidence of service provisioning (e.g., breach of contract, service quality not met, missing items) involving multiple parties has to be collected from heterogeneous computing devices[9] and must safely be recorded in an integrity-assured database.

3.3. RQ3: How to gear up a distributed ledger for enforcing the penalty rules?

The enforcement of SLA-bound penalty rules shall not be (a) tampered with by any involved parties; (b) operated by humans. With distributed ledger technologies now on the rise, we expect to use smart contracts as technological leverage to implement such an enforcement mechanism. A smart contract is essentially a program stored on a blockchain and executed automatically when certain conditions are met. Technically speaking, we need to translate the logic of these penalty rules into the program code of one or more smart contracts.

[9]IoT wearables, QR code/barcode readers, fingertip sensors, etc. in use today make a wide range of technical choices for computerizing service encounter during a service transaction.

4. The PenChain framework

In this section, we first describe the top-level components in PenChain (Subsection 4.1), which is complemented by Subsection 4.5 looking under the hood. In Subsections 4.2, 4.3, and 4.4, we lay the groundwork for the representation of the SLAs, how to combine them to form service bundling and how to rank service offerings by their SLA. The algorithms are presented in Subsection 4.6. In Subsection 4.7, we showcase the chief function of PenChain.

4.1. Overall architecture

As illustrated in Figure 1, we proceed by defining the top-level components of PenChain together with those on the provider's side and the customer's side. The nodes of *Customer-Node*, *Provider-Node*, and *PenChain-Main* each represent either a cloud or a computing server. *Customer-Node* refers to a node hosting the user interface (*Browser*) through which customers request a service registered in PenChain. *Provider-Node* hosts *Browser* and a component that facilitates the service provision (*Providing Services*). As a customer in PenChain might someday become a provider following the rationale of Web3, *Provider-Node* should be backward compatible to *Customer-Node*.

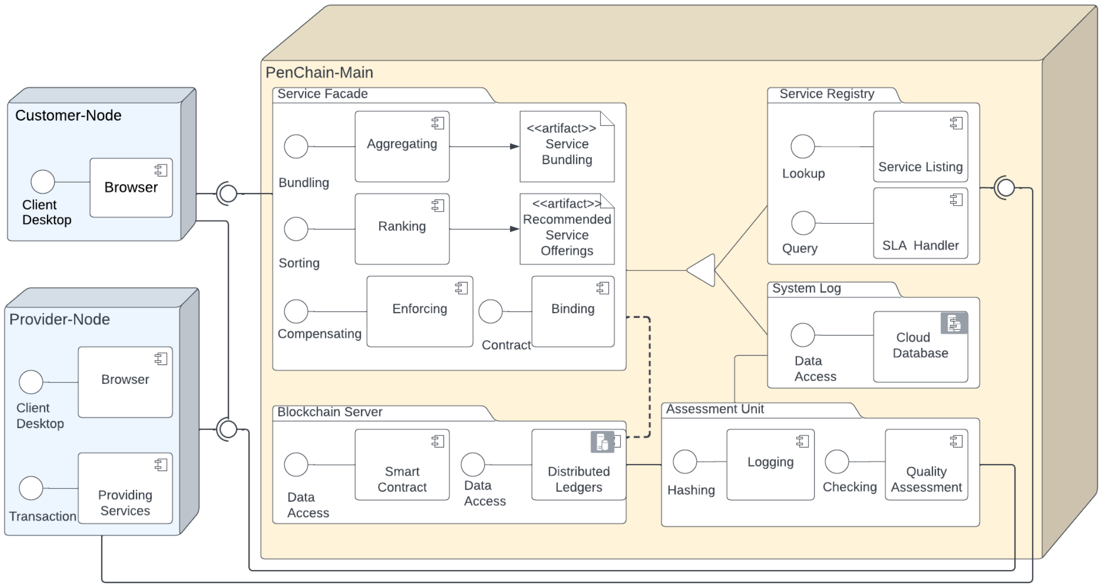


Figure 1: The component view of PenChain's system architecture.

PenChain-Main is the node that hosts the following components of PenChain: service discovery, service log, SLA enforcement, and quality assessment. The component of *System Log*, which captures the log of all service transactions, is connected to *Service Registry* and *Service Façade* via a ternary connector. The component of *Assessment Unit* sends an assessment on the artifacts created during a service transaction to the underlying blockchain, hence a connector that links it to *Blockchain Server*. This assessment is too sent to *System Log* for storing the transactional evidence of the service being consumed. *Service Façade* is the entrance point for the customers to search for individual service offerings or a service bundling. *Service Registry* facilitates the discovery of services and allows their SLA to be queried.

In accordance with the principle of non-interference, as we demonstrate in Figure 1, service delivery actually occurs outside *PenChain-Main*. For instance, an accommodation service is an exchange act that takes place between a hotel and a group of tourists with PenChain's minimal supervision. This concern is even more prevalent in the domain of data provisioning due to data privacy – data exchange takes place between the consumer and the provider without having *PenChain-Main* participative involved. Though *Assessment Unit* in our architecture stores the transactional evidence by invoking components *Logging* (to write down to a cloud database) and *Quality Assessment* (to obtain an assessment of the service outcome), it is nevertheless not participative engaged in any service transaction that is mediated by PenChain.

4.2. Capturing penalty rules to represent a penalty-aware SLA

Our formal reasoning offered in this subsection is twofold: (a) the modeling of the penalty rules, which explicitly addresses research question RQ1; (b) the logic behind comparing the SLAs, which in part addresses research question RQ2.

A penalty rule represents contractual obligations between service providers and consumers. The modeling of these rules requires the use of a language designed for contract modeling that considers them to be business contracts. Business contracts specify obligations, permissions, and prohibitions as mutual agreements between business parties [84], as well as the terms of the breach of contract. In the context of our study, we are more interested in the ways in which the rules are formulated and the distribution of compensation. Deontic operators capture the contractual modality (i.e., obligations, permissions, and prohibitions) [85]. Governatori represents a contractual rule as $r : A_1, A_2 \dots A_n \vdash C$ where each A_i is an antecedent of the rule and C is the consequent. Each A_i and C may contain deontic operators. Specifically, the connective $\circledast$ can therefore be informally read as "failing which". It means that the symbol $\circledast$ can be used to represent a relationship between two compensations. It implies that if the first compensation is not fulfilled, the second compensation will be carried out. $O_{hotel}\alpha \circledast O_{hotel}\sigma$ mandates that a hotel is obliged to make sure that α is brought about. Failure to do so results in a violation, which the hotel can repair by supplying σ. Governatori offers additional formality to reason about the merging of contractual rules in his work.

Example 1. Let's consider the situation when a tourist and her family decided to book a deluxe room at Jade Hotel for their summer holiday. A penalty rule in this accommodation service reads: *"Jade hotel will provide a room that has a balcony with a sea view. In case the hotel is unable to arrange a sea view room as promised, it shall provider her with an alternative room of a luxury interior design, or king size double bed. If this compensation option is unavailable, Jade hotel will offer two adult members of the tourist's family free access to the hotel's spa during their stay. If neither of these compensations could be arranged, the tourist is entitled to a 50% discount on her accommodation at Jade hotel upon checkout."*. We formally express this penalty rule as follows.

$$\boxed{r : \neg seaview \vdash O_{hotel}AltRoom \circledast O_{hotel}FreeSpa \circledast O_{hotel}DiscountWhenCheckout}$$

where the components are expressed in first-order logic in the following.

$seaview \equiv \exists ro \in Jade, t \in Tourist : booked(t, ro) \wedge BalconyWithSeaView(ro)$

$AltRoom \equiv \exists ar \in Jade, t \in Tourist : (LuxuryInterior(ar) \vee KingSizeDoubleBed(ar)) \wedge CheckIn(t, ar)$

$FreeSpa \equiv \exists spa \in Service, ar \in Jade, t \in Tourist : FreeAccess(ar, spa) \wedge CheckIn(t, ar)$

$DiscountWhenCheckout \equiv \exists ro \in Jade,\ t \in Tourist,\ spa \in Service : booked(t, ro) \wedge CheckOut(t, ro) \wedge Discount50Percent(t, ro)$

A semiring is an algebraic structure with two operations. A semiring can be thought of as being similar to a ring, but without the requirement that the addition operation be commutative. The two operations in a semiring are called addition and multiplication. In other words, it is an algebraic structure that generalizes the properties of both a ring and a semigroup [86].

Definition 4.1 (Semiring)**.** A *semiring* is a tuple $\langle \mathcal{A}, \oplus, \otimes, \bar{0}, \bar{1} \rangle$ where

- $\mathcal{A}$ is a set and $\bar{0}, \bar{1} \in \mathcal{A}$;
- $\oplus$, called the *additive operation*. It is a commutative, associative operation having $\bar{0}$ as its neutral element (i.e. $a \oplus \bar{0} = a = \bar{0} \oplus a, \forall a \in \mathcal{A}$);
- $\otimes$, called the *multiplicative operation*. It is an associative operation such that $\bar{1}$ is its identity element and $\bar{0}$ is its absorbing element (i.e. $a \otimes \bar{0} = \bar{0} = \bar{0} \otimes a, \forall a \in \mathcal{A}$). Moreover, $\otimes$ distributes over $\oplus$ (i.e. $\forall a, b, c \in \mathcal{A}$, we have $a \otimes (b \oplus c) = a \otimes b \oplus a \otimes c$).

An idempotent semiring is a semiring whose additive operation is idempotent (i.e., $a \oplus a = a$). This idempotence property allows us to endow a semiring with a canonical order defined as $a \leq b$ iff $a \oplus b = b$. We specify the meaning of operators $\oplus$ and $\otimes$ for reasoning over services in Section 4.3.

In our work, the set denoted as $\mathcal{A}$ refers to the set of SLAs in a certain business domain. We consider $\mathcal{A} = \{\langle x_1, x_2..x_n \rangle\}$ where x_i represents the i-th most important objective of an SLA. In particular, we make a case for the three SLA objectives defined below with the objective of penalty rules being understandably considered the foremost. We acknowledge that different customers might develop a different views on the significance of these SLA objectives. In other words, the SLA objectives should be flexibly sequenced rather than being rigidly sequenced in what we call the customer-centric representation[10] of SLAs. Now, let us define the notions of satisfaction, rule-abiding rate and cost in Definition 4.2, Definition 4.3 and Definition 4.4, respectively.

Definition 4.2 (Satisfaction)**.** Informally, the satisfaction of a service perceived by the end-users is about the reliability and empathy of this service [87]. For business services, the *satisfaction* is formally defined as the ratio of the number of successful service delivery to the total number of service transactions.

Definition 4.3 (Rule-abiding rate)**.** A service is associated with several penalty rules, which are supposed to be respected in run-time. The *rule-abiding rate* (and the *breach rate*) of a penalty rule is defined as the ratio of the number of times the said rule is respected (unrespected) to the total number of times the said rule is kicked off. Formally, we have a *rule-abiding rate* = 1 − *breach rate*. A *rule-abiding rate* of an SLA, or *rule-abiding rate* for short (denoted as r), is defined as the minimum value of rule-abiding rates of all penalty rules concerned.

Definition 4.4 (Cost)**.** Let $C_0 = \{c_1, c_2, ...c_n\}$ be the initial set of values of the cost incurred by services. By *closure* of C_0, we mean the smallest set containing all finite summation of elements in C_0: $C_0^+ = \left\{\sum_{k=1}^{\infty} (c_{i_1} + ... + c_{i_k}) | \, c_{i_k} \in C_0\right\}$. As such, the **cost set** is defined as $C = C_0^+ \cap [0, cost_{max}]$ where $cost_{max}$ is the highest cost that the customer may pay. A *cost* (denoted as c) is a payment for a service, which is an element of set C.

[10]We shall take this customer-dependent representation of SLAs into account when reasoning about the aggregation and ranking of services in the next subsections. This flexible sequence of SLA objectives will translate into the algorithmic dynamics for service ranking in Section 4.6.

4.3. Aggregating services

The practice of aggregating services, colloquially known as service bundling, is exercised in various business domains. The central idea of service aggregation is to offer additional service offerings as packages at a more competitive price.

To proceed with service bundling, for the sake of simplicity, let's assume that the SLAs of all the atomic services is denoted as a set of triples $\mathcal{A} = \{\langle s, r, c\rangle\}$ where rule-abiding is the most important objective, satisfaction: the second and cost: the least. We proceed in aggregating the SLAs of the constituent services. Suppose that we have n services and consider the i-th service for which we write $L_i = \{\ell_{ij} | j = 1, ..., m_i\}$ where ℓ_{ij} represents the the j-th level of its multilevel SLA.

Now, let's put $\alpha^i_{\ell_{ij}}$ to denote the j-th level of the i-th service's multilevel SLA where $i \in [1, n]$; $j \in [1, m_i]$. This construct is in fact a triple $\langle r, s, c\rangle \in \mathcal{R} \times \mathcal{S} \times \mathcal{C}$, where $\mathcal{R}, \mathcal{S}, \mathcal{C}$ are the sets of rule-abiding rates, satisfactions and costs, respectively.

Definition 4.5. To represent the multilevel SLA of service aggregation, we consider a group of aggregated services as a subset of k services $\{so_i | \; i \in I \subset [n], |I| = k\}$ each of which comes with multilevel SLA as exemplified in Table 3. Out of this subset, the cartesian product of the multilevel SLAs of the aggregated services is represented by a 2-dimensional array of size $(\prod_{i\in I} m_i) \times k$, where each element is a triple $\langle r, s, c\rangle$. For each given row consisting of k SLAs in this array, we already know the fixed level $\ell_i \in L_i$ is defined. Hence we write $\langle r^i_{\ell_i}, s^i_{\ell_i}, c^i_{\ell_i}\rangle$, $i \in I$, for the SLAs in this row. Then we define the *combined SLA* (denoted as Ω), over these SLAs as follows.

$$\Omega = \bigodot_{i\in I\subset[n]} \alpha^i_{\ell_i} = \langle \min_i r^i_{\ell_i}, \min_i s^i_{\ell_i}, \sum_{i\in I} c^i_{\ell_i}\rangle. \tag{1}$$

Example 2. We request a group of services as subset $\{so_1, so_4, so_5\}$ (i.e. $k = 3$) and $I = \{1, 4, 5\}$. Suppose that we have $L_1 = \{\ell_{11}$: *Intermediate*, ℓ_{12} : *Advanced*$\}$, $L_4 = \{\ell_{41}$: *Basic*$\}$ and $L_5 = \{\ell_{51}$: *Intermediate*, ℓ_{52} : *Advanced*$\}$. Out of this subset, the Cartesian product of this set is represented by a two-dimensional array of size $(\prod_{i\in I} m_i) \times 3$ as follows.

$\langle r^1_{\ell_{11}}, s^1_{\ell_{11}}, c^1_{\ell_{11}}\rangle$	$\langle r^4_{\ell_{41}}, s^4_{\ell_{41}}, c^4_{\ell_{41}}\rangle$	$\langle r^5_{\ell_{51}}, s^5_{\ell_{51}}, c^5_{\ell_{51}}\rangle$
$\langle r^1_{\ell_{11}}, s^1_{\ell_{11}}, c^1_{\ell_{11}}\rangle$	$\langle r^4_{\ell_{41}}, s^4_{\ell_{41}}, c^4_{\ell_{41}}\rangle$	$\langle r^5_{\ell_{52}}, s^5_{\ell_{52}}, c^5_{\ell_{52}}\rangle$
$\langle r^1_{\ell_{12}}, s^1_{\ell_{12}}, c^1_{\ell_{12}}\rangle$	$\langle r^4_{\ell_{41}}, s^4_{\ell_{41}}, c^4_{\ell_{41}}\rangle$	$\langle r^5_{\ell_{51}}, s^5_{\ell_{51}}, c^5_{\ell_{51}}\rangle$
$\langle r^1_{\ell_{12}}, s^1_{\ell_{12}}, c^1_{\ell_{12}}\rangle$	$\langle r^4_{\ell_{41}}, s^4_{\ell_{41}}, c^4_{\ell_{41}}\rangle$	$\langle r^5_{\ell_{52}}, s^5_{\ell_{52}}, c^5_{\ell_{52}}\rangle$

For each given row consisting of three SLAs, as we already know the fixed level, we therefore have $< r^i_{\ell_i}, s^i_{\ell_i}, c^i_{\ell_i} >$, $i \in I$. Hence, we rewrite the first row in the following.

$\langle r^1_{\ell_1}, s^1_{\ell_1}, c^1_{\ell_1}\rangle$	$\langle r^4_{\ell_4}, s^4_{\ell_4}, c^4_{\ell_4}\rangle$	$\langle r^5_{\ell_5}, s^5_{\ell_5}, c^5_{\ell_5}\rangle$

In this row, we apply Formula 1 to determine the combined SLA and label it Ω_q, where q serves as row indices.

$$\begin{aligned}\Omega_1 &= \langle \min\{r^1_{\ell_1}, r^4_{\ell_4}, r^5_{\ell_5}\}, \min\{s^1_{\ell_1}, s^4_{\ell_4}, s^5_{\ell_5}\}, (c^1_{\ell_1} + c^4_{\ell_4} + c^5_{\ell_5})\rangle \\ &= \langle \min\{93\%, 94\%, 94\%\}, \min\{94\%, 95\%, 98\%\}, (\$5 + \$10 + \$20)\rangle \\ &= \langle 93\%, 94\%, \$35\rangle\end{aligned}$$

We repeat this task for each remaining row in the two-dimensional array above. As a result, Table 1 presents the combined SLAs for the following service aggregation so_1 (Intermediate), so_4 (Basic), so_5 (Intermediate) into Ω_1; so_1 (Intermediate), so_4 (Basic), so_5 (Advanced) into Ω_2; so_1 (Advanced), so_4 (Basic), so_5 (Intermediate) into Ω_3; so_1 (Advanced), so_4 (Basic), so_5 (Advanced) into Ω_4.

Table 1: Service aggregation involves compiling the SLAs of the constituent services.

Label	**Service aggregation**	**Combined SLA** $\langle r, s, c\rangle$
Ω_1	so_1 (Intermediate), so_4 (Basic), so_5 (Intermediate)	$\langle 93\%, 94\%, \$35\rangle$
Ω_2	so_1 (Intermediate), so_4 (Basic), so_5 (Advanced)	$\langle 93\%, 94\%, \$45\rangle$
Ω_3	so_1 (Advanced), so_4 (Basic), so_5 (Intermediate)	$\langle 94\%, 95\%, \$40\rangle$
Ω_4	so_1 (Advanced), so_4 (Basic), so_5 (Advanced)	$\langle 94\%, 95\%, \$50\rangle$

4.4. Ranking

The set of SLAs for a particular business domain is referred to as $\mathcal{A}$ in this work. We write $\mathcal{A} = \{\langle x_1, x_2..x_n\rangle\}$ where x_i represents the i-th most important objective of an SLA. Again, for the sake of simplicity, let's denote the set of SLAs as $\mathcal{A} = \{\langle s, r, c\rangle\}$ where r: rule-abiding rate, s: satisfaction, c: cost. In particular, we argue in favor of the three SLA objectives specified below, with the objective of penalty rules being the one that is, naturally, seen as the most important. We understand that the importance of these SLA objectives may be interpreted differently by specific clients at different times. The SLA objectives ought not to be rigorously sequenced and should have some degree of flexibility in the order in which they are presented in the customer-centric depiction of SLAs.

We recall that PenChain facilitates service aggregation, allowing atomic service offerings to be bundled to form aggregated services. They will be mixed up in a search result, necessitating a polymorphic ranking procedure that sorts them by the SLA regardless if they are atomic, bundled or an aggregation of aggregated services. More precisely, we are in search of an effective comparison technique that works uniformly any SLAs denoted as triples $\langle r, s, c\rangle$.

To devise a comparison technique, we treat $\mathcal{A}$ as a totally ordered set. The total order on a set is a form of ordering the elements of the set, assuming that certain elements precede others, with the understanding that any two elements can be compared in one way or another. Formally, we have $\Omega_i \geq \Omega_j$ if $(r_{\Omega_i} > r_{\Omega_j})$ or $(\mathrm{r}_{\Omega_\mathrm{i}} = \mathrm{r}_{\Omega_\mathrm{j}}) \wedge (\mathrm{s}_{\Omega_\mathrm{i}} > \mathrm{s}_{\Omega_\mathrm{j}})$ or $(\mathrm{r}_{\Omega_\mathrm{i}} = \mathrm{r}_{\Omega_\mathrm{j}}) \wedge (\mathrm{s}_{\Omega_\mathrm{i}} = \mathrm{s}_{\Omega_\mathrm{j}}) \wedge (\mathrm{c}_{\Omega_\mathrm{i}} \leq \mathrm{c}_{\Omega_\mathrm{j}})$.

In order to reason about the SLAs, we utilize the mathematical construct of semiring where the $\oplus$ operation is the max operation concerning this order. The $\otimes$ operator is a multiplication that acts differently on each component of an element in $\mathcal{A}$. The $\otimes$ operator's action on $\mathcal{R}$, $\mathcal{S}$ is taking the minimum. The $\otimes$ operator's action on C is ordinary addition. More precisely, let $a = \langle r_1, s_1, c_1\rangle$ and $b = \langle r_2, s_2, c_2\rangle$, then $a \otimes b$ is defined as follows.

$$a \otimes b := \langle \min\{r_1, r_2\}, \min\{s_1, s_2\},\ c_1 + c_2\rangle$$

Example 3. Table 2 shows an ordered list of aggregated services where Ω_3 ranks first and Ω_2 is the least preferred aggregation. The column to the right of this table prints the combined SLA of the listed aggregated services, explaining why they are ranked first, second, etc. as shown in the leftmost column.

Table 2: Service aggregations are sorted by their combined SLA in the same way service offerings are

Rank	**Label**	**Service aggregation**	**Combined SLA** $\langle r, s, c \rangle$
1	Ω_3	so_1 (Advanced), so_4 (Basic), so_5 (Intermediate)	$\langle 94\%, 95\%, \$40 \rangle$
2	Ω_4	so_1 (Advanced), so_4 (Basic), so_5 (Advanced)	$\langle 94\%, 95\%, \$50 \rangle$
3	Ω_1	so_1 (Intermediate), so_4 (Basic), so_5 (Intermediate)	$\langle 93\%, 94\%, \$35 \rangle$
4	Ω_2	so_1 (Intermediate), so_4 (Basic), so_5 (Advanced)	$\langle 93\%, 94\%, \$45 \rangle$

4.5. Under the hood of PenChain

Figure 2 illustrates the inner workings of *PenChain-Main* – a node where the components of PenChain excluding the provider-side and customer-side computing are deployed (see Figure 1). We utilize the layered architecture style to describe how components could be developed and deployed in tandem with the smart contracts that update the trusted ratings of the service providers and enforce the penalty rules in PenChain. They are organized into three layers, namely *Business Layer*, *Persistence Layer* and *Blockchain Layer*. Let's elaborate them in the following.

4.5.1. Business layer

The layer to the left of Figure 2 represents the business logic of PenChain. It is organized into four groups. Group 1 is made of interconnected modules that interact directly with the providers and give some input to the other modules in subsequent layer to serialize the service transactions: *Services*, *SLAs* and *Penalty Rules*. Group 2 (*Aggregator* and *Ranking*) assists the customers in locating a service offering or an aggregated service by computing the SLA parameters in accordance with Definition 4.5. The modules in this group send requests to the next layer (*Persistence Layer*) to perform data queries on the cloud database. Customer engagement is carried out by Group 3, which is comprised of the following modules: *Search* and *Contract*. In addition to allowing the customers to view the contractual specifics of the service they are bound to, including its SLA, the *Contract* module makes any in-progress compensation attempt to fix a breach of contract, which could be activated by invoking *Penalty-Aware SLAs*, transparent to them. The key modules in Group 4 invoke the next two layers to write down the service's transactional evidence to both the cloud database and the blockchain.

4.5.2. Persistence layer

As the name suggests, the layer situated in the middle of Figure 2 helps PenChain serialize service transactions that were executed. *General* handles the transactional log of a service that is not highly confidential (e.g., location, timestamp, service participants who were engaged in the transaction, the actual SLA that was associated with the service). *Logs* stores more sensitive information about the service (e.g., quality experienced by the customers and whether it falls below or surpasses a critical threshold value). Both of them write the service's transactional records they receive from Group 1 and Group 4 of *Business Layer* down to a cloud database (our pick here is Azure SQL Database).

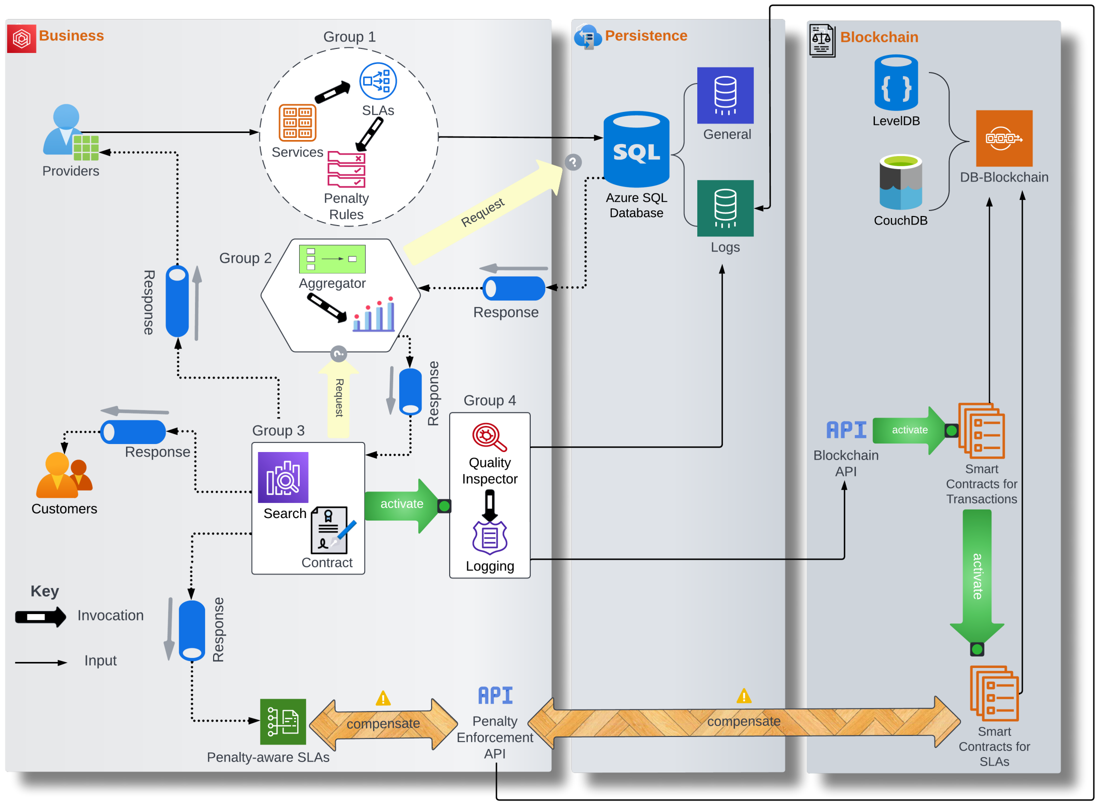


Figure 2: PenChain utilizes replaceable and loosely-coupled units for keeping track the service transactions and enforcing the penalty-aware SLAs. They are conceptually organized into three layers.

4.5.3. Blockchain layer

The rightmost layer depicted in Figure 2 is dedicated to the underlying distributed ledger and essential smart contracts of PenChain. It is designed to store immutable records as successive blocks, each representing a service transaction. Technically, we install a private blockchain using Hyperledger technologies on our cloud. We refer to Hyperledger's storage capacity as *DB-Blockchain*, which is made of two separate data sources: an internal*LevelDB*[11] and an external *CouchDB*[12]. The former holds chaincode data as key-value pairs, while the latter facilitates sophisticated data queries in the chaincode.

[11]LevelDB `https://github.com/google/leveldb` is an open source database system that is based on concepts from Google's Bigtable database system. It could be embedded in the peer process of a blockchain, providing low-level key-value storage and retrieval capabilities.

[12]Apache CouchDB `http://couchdb.apache.org` is an open source database that uses a schema-free cross-platform data model.

Logging in *Business Layer* submits the assessment of a service transaction and the corresponding hash in Azure SQL Database to *Smart Contracts for Transactions* in this layer through *Blockchain API*, activating *Smart Contracts for SLAs* in order to reassess the SLA's satisfaction. Upon receiving this request, the module submits a compensation request to *Penalty-aware SLAs* via *Penalty Enforcement API* in *Business Layer* if the assessment falls short, indicating an instantaneous breach of contract. Now, the control is granted to the modules in *Business Layer* again. Acceptable indemnification criteria have been incorporated into *Contract*. In effect, *Penalty-aware SLAs* monitors the compensation workflow that has already been activated and sends a confirmation to *Smart Contracts for SLAs* to update the SLA's rule-abiding rate. To illustrate that there are requests and confirmation that go back and forth between the involving modules, we put two 2-way block arrows to express this compensation workflow in Figure 2.

4.5.4. Programmatically enforcing penalty rules

A smart contract, also known as a chaincode in Hyperledger, is a computer program that defines the rules and conditions for executing transactions. In this subsection, we walk through such a chaincode to illustrate why the enforcement of penalty-aware SLAs is attainable given today's distributed ledger technology. In other words, we program the penalty rules to automate the SLA enforcement. The chaincode is written in a rather straightforward fashion in accordance to the general structure of the penalty rules expressed in deontic logic. Nevertheless, to fully implement a penalty rule, we programmatically rely on replaceable modules that are external to the blockchain to take reparation action on behalf of the service provider whose the penalty rule in question is being fired to fix this temporary breach of contract.

```go
func (t *CPRChaincode) callPenaltyRule(stub shim.ChaincodeStubInterface,
    attributes []string) cp.Response {

    comp, err := strconv.Atoi(attributes[0])

    ContractID := attributes[1]
    ProviderID := attributes[2]
    RuleID     := attributes[3]

    // the following array captures a list of links pointing to external
    APIs,
    // for example, in the case of the aforementioned accomodation service,
    // {"https://provider.com/api/alt-room",
    //  "https://provider.com/api/free-spa",
    //  "https://provider.com/api/discount"}
    // this array is fetched from the ledger
    apiList := getExternalAPIList(ContractID, RuleID)

    var resp *http.Response
    var api string
    var status = false

    for comp < len(apiList) {
        api = apiList[comp]
        resp, err = http.Post(api, "application/json", bytes.NewBuffer([]
    byte(ContractID)))
        if err != nil {
            comp++
26         } else {
27             status = true
28             break
29         }
30     }
31 
32     calRuleAbidingRate(ProviderID, RuleID, status)
33     return shim.Success(nil)
34 }
```

Listing 1: Chaincode on Hyperledger Fabric that implements a generic penalty rule

Let's turn our attention to Example 1 again while walking through the chaincode given in Listing 1. We declare an entry-point function under the name `callPenaltyRule`, of which arguments are wrapped in an array of strings called `attributes`. Let's recall that a typical penalty rule represented in deontic logic contains a series of antecedents and a consequent. To be able to programmatically enforce the said penalty rule, we first fetch list of external APIs corresponding these deontic components (at line 15 of Listing 1) and store them in an ordered array named `apiList` by calling a user-defined function named `getExternalAPIList`. The code comments that are put before this function call (lines 9-14) exemplify what the array holds for Example 1. Note that the chaincode presented here is generic in the sense that it is programmed to trigger any compensation modules that have been readily deployed by the service provider involved in the contract (identified by `ContractID`). More specifically, the providers are expected to make the relevant compensation available via an API[13] when they published their penalty rules. The function then attempts to post a request to an API call module in `apiList` (at line 23 of Listing 1) until a compensation is confirmed. If an error occurs during the request owing to, for instance, technical failure or the service provider's inability to compensate (for example, a situation where a hotel has no compensation options to make up for a tourist who checked in a non-oceanfront room). If no post requests are confirmed at all, which signifies that the penalty rule in question has not been adhered to, `status` still holds the boolean value of `false` after the loop is terminated. At line 32 of Listing 1, function `calRuleAbidingRate` is utilized to recalculate the rule-abiding rates of the involved service provider (identified by `ProviderID`) to write down whether the penalty rule in question (identified by `RuleID`) has been respected or not (indicated by `status`).

4.6. *Ranking algorithms*

We propose algorithms needed for ranking a set of service offerings, allowing the search result to be sorted that best matches the customers' preference. Service offerings are mixed up in a search result – some of them might be of the same function but are offered by different providers. Moreover, an element in this list could either be an atomic service or a service aggregation. The proposed algorithms materialize our groundwork on the semiring presented in the previous subsections. The main idea is to devise Algorithm 2 as a comparator function that will be fed into a sorting function (Algorithm 1).

[13]The compensation module might be fully automatic (e.g., discount). In many cases, it is operated by human (e.g., arranging an alternative room). The API at which this module is called from the chaincode is safely and persistently stored in the ledger.

Algorithm 1: Sorting a search result by the SLAs taking into account the customer's preference.

```
Data: $L_{serv}$: a search result that needs to be sorted;
1  Begin
2      if the service customer has not interacted with PenChain yet then
3          $L_{objective} \leftarrow$ the PenChain's default list of SLA objectives;
4      else
5          $L_{objective} \leftarrow$ customized list of SLA objectives;
6          Sort $L_{objective}$ in a descending order by the number of views received
             for each objective;
7      end
8      foreach $serv \in L_{serv}$ do
9          if serv is a service aggregation that comes with the SLA then
10             Compute the SLA of serv in accordance with (1) in Definition
                 4.5;
11         end
12     end
13     sort $L_{serv}$ using the comparator function defined in Algorithm 2 and
         $L_{objective}$;
14 End
```

L_{serv} in Algorithm 1 denotes a set of service offerings listed in a search result. We recall that the SLA objectives are flexibly sequenced according to the customer's preference. $L_{objective}$ is an ordered list that determine the relative importance of constituent objectives in the SLA through the lens of the service customer. For example, if the rule-abiding rate matters most to a certain customer, the corresponding item is placed first in her $L_{objective}$. This list needs to be re-sorted in descending order when PenChain records a substantial amount of additional view counts, suggesting that the service customer has re-prioritized the SLA constituent objectives in their mind. The algorithm is designed to work with an unlimited number of constituents though in this work we assume that $L_{objective}$ is a list of three. The algorithm addresses two possibilities: a newly registered customer who has yet to interact in PenChain (e.g., to submit a search request, to view different service offerings listed in a search result) and a returned customer whose viewing history is kept track of her profile. Note that this algorithm treats all elements of L_{serv} in the same way, no matter whether they are atomic or aggregated.

Algorithm 2: Comparing a pair of service offerings by their SLAs.

```
Input:
$sla_1$, $sla_2$: A pair of SLAs to be compared;
$L_{objective}$: a list of objectives sorted in descending order by the customer's view count;
Output: +1 if $sla_1$ is better than $sla_2$, 0 if equal, -1 if worse;
Begin
    foreach $f_{obj} \in L_{objective}$ do
        if MAX is the optimization function for $f_{obj}$ then
            if $sla_1[f_{obj}]$ is greater than $sla_2[f_{obj}]$ then
                return 1;
            else
                if $sla_1[f_{obj}]$ is less than $sla_2[f_{obj}]$ then
                    return -1;
                else
                    continue;
                end
            end
        else
            if $sla_1[f_{obj}]$ is less than $sla_2[f_{obj}]$ then
                return 1;
            else
                if $sla_1[f_{obj}]$ is greater than $sla_2[f_{obj}]$ then
                    return -1;
                else
                    continue;
                end
            end
        end
    end
    return 0;
```

Algorithm 2 acts as a comparator function that determines if a given service (sla_1) should be placed above or below another (sla_2) in a search result. The list represented by $L_{objective}$ is literally passed from Algorithm 1, telling which SLA objective is the foremost criterion, the second most important, and so on. To make this algorithm generic, in the main loop, we iterate through all elements of $L_{objective}$, not necessarily confined to the three SLA objectives as we make the case for in this paper. We assume that each objective is associated with an optimization operation, either maximization or minimization, advising us to retain the order (e.g., comparing the satisfaction) or reverse it (e.g., comparing the costs) when comparing. The loop continues down the list of $L_{objective}$ until we can figure out if sla_1 is preferred over sla_2 or vice versa. This algorithm returns zero if it goes through all objectives in $L_{objective}$ without decisively determining the order between sla_1 and sla_2, suggesting that these two services should be placed at the same position in the search result.

4.7. *Multi-criteria search*

In this section, we showcase the advanced search feature of PenChain that allows the customers to search for and locate right service offering. To enable this service search feature, the providers are supposed to make their service details and service location[14] available in the service registry of PenChain.

As illustrated in Figure 3, a text box with auto-complete in the right pane allows the service's geographical location to be entered. Suggestions of the location in this text box are sourced from Bing Maps for the customer's convenience. The search result is visually displayed on the map below in the left pane, with service offerings are shown under the same teardrop-shaped icon but at various locations. Each of them comes with a tooltip that pops up when the customer clicks on it, displaying its name and further details of the service being represented. To show an example, on the map there is a teardrop-shaped icon that is yellow in color representing a data service that reports the water's pH levels in a farming site. The customer might find other widgets in the tooltip useful for viewing the SLA-bound historical values of the service she is looking at, which is instrumental in improving the customer's comprehension of the rating of its service provider.

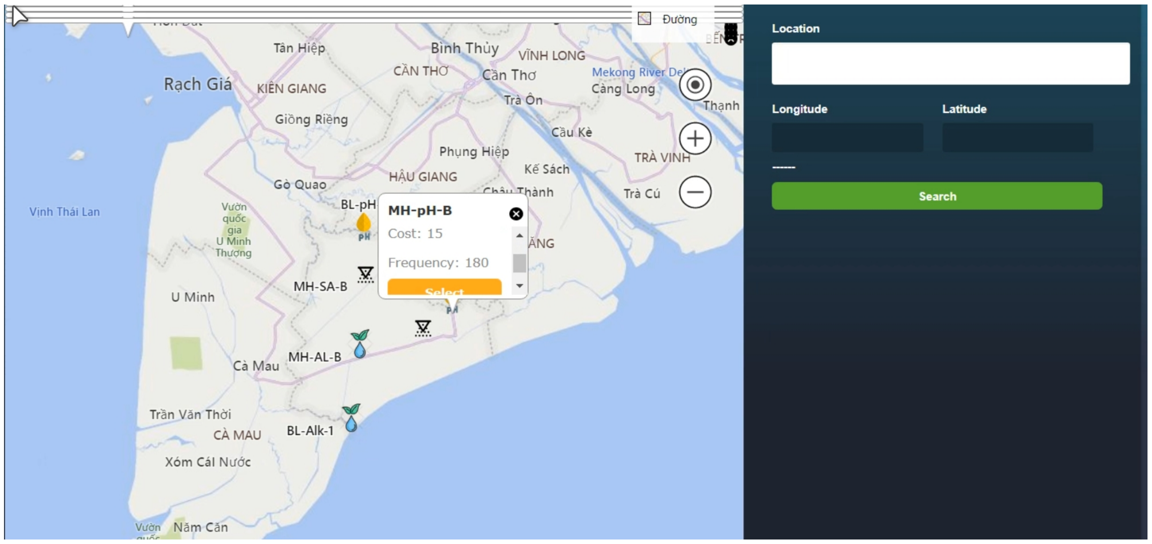
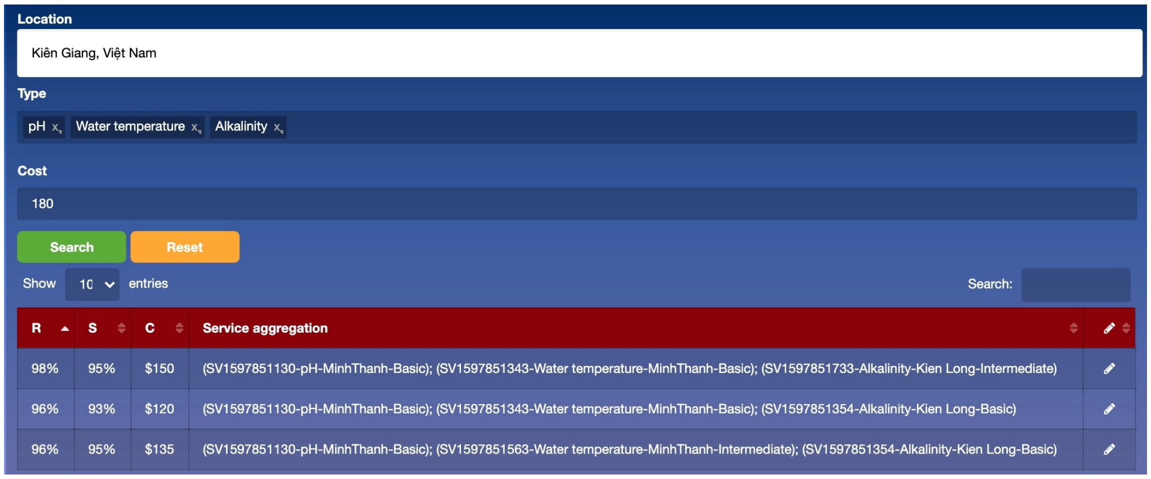


Figure 3: Service offerings in a search result are visually shown on Bing Maps, offering the service customer a spatial sense when examining them prior to a possible service binding.

An alternative user interface for multi-criteria search is depicted in Figure 4. The search dialog in this user interface has several text boxes where the customer enters the keywords, expected costs, and of course the service's location. The search result is then shown in the table below, which is sorted according to Algorithm 1. The text box for entering the service's location in Figure 4 comes with auto-complete fed from Bing Maps just like its counterpart in the search dialog of Figure 3.

The service customer may toggle the search mode at any time as she interacts with PenChain. In both modes, she should be able to pick a service offering from the search result to proceed further (e.g., adding it

[14]One should not get confused between the actual location of the service in question, e.g. where sensors are installed for data provisioning, and the postal address registered in the profile of the organization or agent who provides this service.


Location

Kiên Giang, Việt Nam

Type

pH, Water temperature, Alkalinity

Cost

180

R	S	C	Service aggregation	
98%	95%	$150	(SV1597851130-pH-MinhThanh-Basic); (SV1597851343-Water temperature-MinhThanh-Basic); (SV1597851733-Alkalinity-Kien Long-Intermediate)	
96%	93%	$120	(SV1597851130-pH-MinhThanh-Basic); (SV1597851343-Water temperature-MinhThanh-Basic); (SV1597851354-Alkalinity-Kien Long-Basic)	
96%	95%	$135	(SV1597851130-pH-MinhThanh-Basic); (SV1597851563-Water temperature-MinhThanh-Intermediate); (SV1597851354-Alkalinity-Kien Long-Basic)	

Figure 4: A tabular view of a search result that allows the service customer to comprehensively examine service offerings listed ion a search result before choosing one of them.

to a service shopping cart, checking out to start the service contract).

5. Scenario-based evalution: case of high-precision agriculture

We describe the need for enforcing the SLAs of data provisioning in smart shrimp farming in Section 5.1, analyze its challenges in Section 5.2 and present an engineering approach for it in Section 5.3. Our goal is to demonstrate the relevance of our PenChain framework to the enforcement of penalty-aware SLAs in a specific business domain as a case study. To make this section self-contained, we explain the underpinning technologies in Section 5.4.

5.1. The bad

The Lower Mekong River Basin in Vietnam has a unique physical terrain with interlaced ditches and canals. The region's terrain is well-suited for farming and fishing, making agriculture a significant part of its economy. This business sector comprises corporations, private shrimp farming families, and many other IoT players. In light of precision agriculture, the productivity of shrimp farming depends on water quality, water temperature, dissolved oxygen, etc. necessitating the monitoring and automated regulation of these parameters. Unfortunately, data gathering is an error-prone and costly technical process, particularly when collecting data sets on the sparsely spread shrimp ponds in the region. With the rise of the Internet of Things, agricultural practitioners and data engineers expect to benefit from next-gen data provisioning and service engineering.

5.2. The ugly

As more and more data service offerings become available to these end users, a novel approach to service registry and discovery should be devised to let them gain access to the right offerings that best match their intention. PenChain should allow both in-house and cloud-based IoT data providers to join and provision their real-time farming datasets. End-users[15] who wish to consume a farming data package will submit a location-based, schema-specific search request to this platform (e.g., looking for all service providers that report the pH levels of all farming sites over a province), for which the search result needs to be sorted – the closer to the top of this search result a provider is placed, the higher reputation score it has from the requesting end-user's perspective.

Introducing modern IoT technologies to agriculture would create an opportunity for farmers, agricultural engineers, and data experts to fruitfully collaborate in an ecosystem. PenChain could be tailored to facilitate exchange of farming data following the so-called data-as-a-service strategy. The data services registered in PenChain might be invoked from a custom software product developed in a DevOps context.

5.3. The good

In Table 3 we populate data services being offered in PenChain. Each service is uniquely identified by a label (so_1, so_2, etc.) and referred to by a short name. The service providers are anonymously denoted as P^1_{cloud}, $P^2_{in-house}$, $P^3_{in-house}$ and P^4_{cloud}. To the right of Table 3, we describe the contractually formulated, multilevel commitments between these service providers and the end-users. We show no more than three levels for each of these service agreements to keep it simple. The basic, intermediate, and advanced levels are with a logical progression to monthly costs and data frequency. Geographically speaking, P^1_{cloud} is located in Giang Thanh, $P^2_{in-house}$ in Kien Luong, $P^3_{in-house}$ in Hon Dat, P^4_{cloud} in Ha Tien. These proper names refer to the rural districts of Kien Giang Province in the Mekong Delta of Vietnam.

Table 3: PenChain facilitates the provisioning of IoT data services in the Mekong Delta

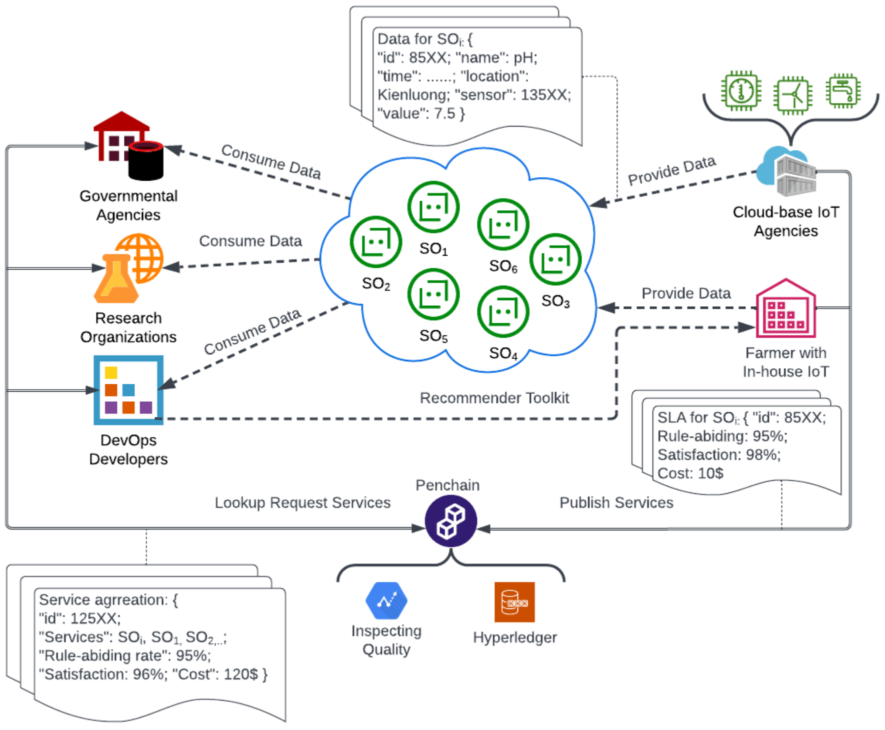
Label	**Data Services**	**Provider**	**Multilevel**	**Data Frequency**	**Cost**
so_1	Water temperature	P^1_{cloud}	Intermediate	Every 60 secs	\$5
			Advanced	Every 36 secs	\$10
so_2	Water pollution	P^1_{cloud}	Basic	Every 18 secs	\$10
			Intermediate	Every 36 secs	\$20
			Advanced	Every 25 secs	\$25
so_3	pH	$P^2_{in-house}$	Basic	Every 35 secs	\$3
so_4	Alkalinity	$P^3_{in-house}$	Basic	Every 43 secs	\$10
so_5	Salinity	P^4_{cloud}	Intermediate	Every 40 secs	\$20
			Advanced	Every 25 secs	\$30
so_6	pH	P^4_{cloud}	Intermediate	Every 20 secs	\$10

Service providers $P^2_{in-house}$ and $P^3_{in-house}$ are farmers who deploy IoT sensors in their farming sites that are located in one of the above-mentioned rural districts. They invest in IoT-enabled equipment to monitor their crop and optimize their agricultural production. They rely on on-site sensors to provide the end-users with instantaneous readings of the pH and the alkalinity over a localized area. Due to the lack of computing power and storage, readings of the pH and alkalinity in the past might not come in handy. In contrast, the other two

[15]They are research institutions, corporations, and agricultural extension organizations working on digitized solutions that offset the effect of climate change in the aquatic environment.

providers – denoted as P^1_{cloud} and P^4_{cloud}, harness cloud computing to provide the end-users with enriched data (e.g., spatio-temporal extrapolation) about water temperature and water pollution over a relatively large farming site, which they do not own. Hi-tech agencies like P^1_{cloud} and P^4_{cloud} secure the appropriate permission from the farm owners to operate their IoT devices, networking infrastructure, and cloud servers in one of their farming areas. Their business is to collect large datasets of agricultural production and disseminate them to earn money.

As shown in Figure 5, PenChain allows end-users to request data services offered by the aforementioned service providers. End-users in this scenario encompass DevOps developers, research organizations, and governmental agencies. They make use of the service offerings listed in Table 3 to compose service-oriented applications, for instance combining services so_1 and so_2. Thanks to PenChain, the research organizations can obtain agricultural data for scientific purposes by invoking, e.g., services so_3 and so_6. The governmental agencies rely on numerous data packages obtained via services so_4 and so_5 to fine-tune the agricultural extension for fisheries development in the area. To give a sense of service ranking, a research lab interested in pH readings would prefer so_6 to so_3 because the former comes with a favorable agreement level than the latter.


Data for SO$_i$: {
"id": 85XX; "name": pH;
"time": ......; "location":
Kienluong; "sensor": 135XX;
"value": 7.5 }

SLA for SO$_i$: { "id": 85XX;
Rule-abiding: 95%;
Satisfaction: 98%;
Cost: 10$

Service agrreation: {
"id": 125XX;
"Services": SO$_i$, SO$_1$, SO$_2$,..;
"Rule-abiding rate": 95%;
"Satisfaction: 96%; "Cost": 120$ }

Figure 5: An aggregator business model for services in PenChain.

Over time, as the DevOps developers who compose a service-oriented application for water monitoring will ask for additional data services that are not listed in Table 5. One way to meet this demand is to aggregate functionally-related data services (e.g., both so_1 and so_2 provide data readings in the water) to provision more sophisticated data packages of a farming site (e.g., reporting on the water temperature and water conductivity of the same shrimp pond). Speaking of service-oriented programming, so_1 and so_2 together as a service aggregation would programmatically come in handy. Aggregating so_1 and so_2 not only involves mixing their water-related data and results in their multilevel SLAs being compiled and rationalized.

5.4. Underpinning technologies

In this section, we discuss the major difficulties associated with the adoption of Blockchain technologies and DevOps tools in making data provisioning possible and enforcing any penalty-aware SLAs associated with it. To understand how PenChain facilitates data exchange in smart farming and guarantees that the agreements stated for data provisioning are followed through on, let us look under the hood.

5.4.1. MS Azure

We emulate the applications in use for both the service providers and the customers on Azure[16]. This platform comes with a built-in monitoring capability for Web services, which we found useful for stress testing and deploying concurrent emulations. As for data handling, PenChain utilizes cloud-based relational database services provided by Azure SQL that perform relational queries, searching, and data synchronization.

5.4.2. Blockchain

Ethereum and Hyperledger Fabric are two popular blockchain platforms that provide a variety of functions. The permissionless mode of Ethereum is at the expense of privacy, performance, and scalability. Hyperledger Fabric eliminates this problem by running as a permission blockchain with flexible, extensible consensus and access control techniques. According to [88], public blockchain networks are not suitable for launching the decentralized program for a B2B solution. Businesses may cooperate with distributed ledger developers to launch B2B and cross-industry applications using Hyperledger. We follow this rationale and leverage the smart contracts – computer programs that run on the blockchain when specific criteria (e.g., temporal) are met, in automating the compensation clauses scripted in the SLAs of the data services being offered in PenChain.

5.4.3. Keeping data logs and blockchain in sync

From the viewpoint of software architecture, PenChain is composed of several components. One of them handles the data log that keeps track of all data exchange transactions, implemented using Azure SQL cloud database. As Hyperledger is not inherently for handling big data at high velocity, we store the coarse-grained representation of these transactions in our blockchain database. Thanks to a hashing technique, we should be able to trace each transaction stored in the blockchain back to the corresponding log item in Azure SQL that records the full evidence of the same data exchange transaction. For the sake of dispute resolution, it is always possible to fully retrieve the evidence of any data transaction recorded in PenChain's blockchain for a full-scale investigation.

[16]Microsoft Azure is a cloud computing platform and web portal that lets software developers access and manage cloud resources and services including Web services and data analytics `https://azure.microsoft.com`

5.4.4. *Programming the smart contracts*

Smart contracts written and deployed in PenChain's blockchain serve two purposes. First, they recalculate the accumulated rating of a data service provider upon receiving a confirmation of the quality of data exchanged or a breach of the SLA contract. A special module of PenChain that automatically assess the quality of exchanged data will record any irregularity of data exchange caused by sensor malfunctions, connection timeout, data handling problems, etc. Such an issue would trigger a smart contract that updates the rating of the service provider involved in this data transaction. Second, special smart contracts in PenChain – illustrated in Listing 1, are programmed to fire workflow actions scripted in a penalty rule to trigger the compensation workflow upon detecting an SLA violation.

6. Analytical considerations

In this section, we first discuss how PenChain aids in the big picture of service engineering (Subsection 6.1). In our experiments, PenChain was specifically geared up for precision agriculture (see Section 5 for an in-depth engineering analysis). The scalability of PenChain is discussed separately in Subsection 6.2. In Subsection 6.3, we will remark on the relevance of PenChain to a wider range of business domains.

6.1. *Service publishing, discovery and binding*

Service-oriented computing involves a wide range of phases with service publishing, service discovery and service binding being the major phases. We point out to what extent PenChain would address these engineering phases. To make the case for PenChain's usefulness for service provisioning, we set up eight client apps on a cloud that is separated from our MS Azure cloud where PenChain was deployed.

Service publishing. We follow the common architecture of service-oriented computing by which penalty-aware services should be made available to registered customers of PenChain. Any client application (i.e., software that allows a customer to consume services in mutually trusted ways via PenChain) linked to our platform shall access this service registry without any restriction. Our SLA-oriented service aggregation technique helps enrich the search space in PenChain, allowing more service offerings to be listed in a search query. We require that all providers publicize their penalty rules when joining our platform. Unfortunately, it is a shortcoming of PenChain that we do not fully establish a technical workflow detailing how to publish penalty-aware services.

Service discovery. One of primary function of PenChain is reputation-driven service lookup, i.e. how service offerings are sorted in a search result depends in part on the reputation of their provider. As can be seen in Figure 3 and Figure 4, this function has an intuitive user interface where the customers may view their search results either as: (a) a set of offerings matching the search criteria, which are sorted by their SLA; (b) a filtered list of service offerings displayed on Bing Maps. The former helps the customer comprehend her search result by displaying the all service offerings together with service aggregations while the latter is more intuitive and gives a spatial sense of the service lookup mechanism in PenChain.

Service binding. PenChain was built with modality in mind by which modular clusters will communicate with each other through APIs. A unit that assesses the satisfaction is located in the same cloud as the "main" of PenChain, which will be invoked in every service transaction[17] via a designated API. The "main" of PenChain then triggers a smart contract to update the provider's rating according to the confirmation returned from the assessment unit. As such, PenChain consists of loosely-coupled units that are considered

[17]The invocation will be redirected to another unit that assesses the practicality of the service consumed as if it was in the plug-and-play mode when changing the service domains, e.g., switching from precision agriculture to air quality monitoring.

to be replaceable. To this end, service binding is with minimum intervention from PenChain, letting the service transactions run their course. Yet PenChain is able to enforce publicized penalty rules (and update the provider's rating accordingly) associated with a service transaction if the situation is not going in an unintended workflow.

6.2. Scalability of PenChain in precision agriculture

Suppose PenChain harbors n service providers each of whom will at least launch m data service offerings. There could be several variants of a service offering that differ on the SLA. Let k be the number of times a service offering is bound to customers in PenChain (the more customers registered in PenChain the higher k). Interviewing local farmers in farming sites where we obtained experimental datasets suggests that the data exchange frequency in precision agriculture shall be no more than four times per day. Hence, the total number of service transactions per day can be written as $G = n \times m \times k \times 4$, or $n \times m \times k \div 360$ transactions per minute. Let's analyze the load sustained by PenChain in a few typical scenarios as follows.

- *Provider dominance*. Suppose there are 1000 providers registered in PenChain for the business of farming data provisioning. They each exhibit nine data service offerings. If a service offering is bound to at least nine customers, PenChain processes 225 transactions per minute on average.
- *Customer dominance*. PenChain attracts a large number of customers that frequently request data service offerings from a relatively small number of providers ($n = 100$). Each provider offers nine service offerings, each of which is bound to 50 customers. PenChain processes around 125 transactions per minute in this scenario.
- *Balanced & lightweight*. For example, there are only five providers who each publish 15 data services. The service offerings are bound to at least three customers. PenChain sustains a light load in this scenario – only one transaction per minute.

In the first two scenarios, namely provider dominance and customer dominance, regardless of what clouds the data services and customer applications shall be deployed to, we need to upgrade the server that hosts our Hyperledger Fabric, which is currently the bottleneck of PenChain as the private blockchain database is notoriously unresponsive under high load. We note that the cloud database counterpart of MS Azure is rather elastic, necessitating a smart hashing mechanism that links a transaction in the blockchain and its extended log in the cloud database. PenChain would scale up in the third scenario.

As analyzed in Section 4, PenChain consists of the following loosely coupled components: data services deployed either on the provider's cloud or MS Azure, customer applications, the "main" and the underlying blockchain. In all scenarios, the scalability of PenChain depends on computing resources allocated to the providers' cloud and the Azure cloud for its "main" as well as the responsiveness of Azure SQL database and Hyperledger Fabric in use.

6.3. The relevance of PenChain

Let us point out how relevant PenChain is to several business domains. In general, our proposed framework is suitable for domains where service encounter is intermittent. Too frequent a service encounter model (e.g., in traffic monitoring) would overburden the underlying private blockchain of our Web3-inspired architecture.

6.3.1. Smart farming

"Farmers desperately need access to new technologies in order to compete on world markets. Innovation is crucial."

– Daniel Azevedo, Director Commodities, Trade and Technology at Copa-Cogeca

Data exchange as a service is a common saying in present-day's smart farming. For instance, HARA[18] is a blockchain-based platform for data exchange in the food and agriculture sector. Agdatahub[19] operates trusted platforms for secure data exchange. Centre for Data Sharing[20] makes a prominent use case of data sharing in agriculture.

In the Mekong Delta region of Vietnam, we need a platform where both in-house and cloud-based IoT data providers could join and provision their real-time farming datasets. End-users[21] who wish to consume a farming data package will submit a location-based, schema-specific search request to this platform (e.g., looking for all service providers that report the pH levels of all farming sites located in a given province of Vietnam), for which the search result needs to be sorted – the closer to the top of this search result a provider is placed, the higher reputation score it has from the requesting end-user's perspective.

As more and more data service offerings become available to these end-users, PenChain's service registry and discovery would let them gain access to the right offerings that best match their intention. Penalty rules captured in the SLA of an IoT data service may state what compensation the customer is entitled to (e.g., discount, a data provisioning package for free) but should not be unreasonably one-sided.

6.3.2. Tourism

"...Customers will become empowered through more choice and control..."

– The Travel And Tourism Industry By 2030 (Forbes on Dec 27, 2021)

Establishing credibility between service providers and visitors are vital for the success of tourism sector. Quite often, arguments occur while visitors are in the middle of a tour or service, at the conclusion of which many of them finally find themselves in an undesirable scenario. In a few other circumstances, the firm conducting this trip falls victim to constant, baseless complaints created by the guests. Explicitly structuring contractual agreements could safeguard both of them, as shown in the automobile rental business.

Feedback left on online reputation systems is notoriously known for being biased [7] and unfair. Even when done properly, leaving feedback is a time-consuming exercise that often is ignored by tourists and passengers alike despite being reminded a couple of times after going home from their trip. Having SLAs digitized and enforceable is thus crucial for maintaining the rigor of many online reputation systems in tourism sector. We believe PenChain is customizable for this sector. The SLA specified for tourism services shall include the following obligations and factors.

- penalty rules and rule-abiding rate, which is the ratio of the count of rule-abiding transactions to the total transaction count
- satisfaction rate, which is the ratio of the count of as-described transactions to the total transaction count

[18]`https://www.hara.ag`

[19]`https://agdatahub.eu`

[20]`https://eudatasharing.eu/`

[21]They are research institutions, corporations, and agricultural extension organizations working on digitized solutions that offset the effect of climate change in the aquatic environment.

6.3.3. *Car rental*

"...As blockchain technology drives more opportunities and decreases individual costs for mobility, traditional car ownership is disincentivized..."

- Richard Ma, CEO of Quantstamp, How Blockchain Is Driving New Opportunities In Mobility (Forbes on Mar 11, 2020)

Service-oriented enterprise engineering essentially involves service specification, design and operations. The SLAs between the enterprise in question and its customer should be defined during service specification. While an SLA might be formulated for entities that are external to the said enterprise, it actually is fulfilled by its internals. Let us consider rent-a-car as a service offered by a car rental company for which an SLA states that the customer's waiting time for picking up her rental car at a counter of the company should be no more than fifteen minutes. The fulfillment of this SLA depends on the availability of the people working at the counter and the company's supporting processes (e.g, servicing the car fleet to maximize the number of cars that are readily available for rent). Another SLA specified for this service is to make sure the renter will be given a rental car of the same class specified in her rental reservation (e.g., economy, mid-size). Otherwise the renter should be allowed to pick up a car of a higher class without any surcharges.

As the costs of owning private cars continue to increase in many countries, leasing instead of buying cars has become an economically driven choice. Digital car rentals are on the rise thanks to the advancement of sensor technologies for intelligent transportation systems. In car rental industry, penalty rules – being articulated as reasonable two-sided statements to protect the rights of both the company and the renters, are ever more automated using present-day's sensor technologies. From the administration perspective, a rental company could keep track of almost every movements of and any services done to its car fleet.

7. Future directions

In the future, we will investigate ways of automatically transforming the textual description of penalty rules into deontic logic. The antecedents and the consequent expressed for a penalty rule suggest what API call modules that programmatically support the compensation workflow shall be specified, developed and deployed on the service provider's side. While the chaincode that implements the compensation workflow remains unchanged should a penalty rule is revised, the the corresponding external API call modules should be reprogrammed and redeployed. Work is currently underway to devise a tool proposal that allows registered service providers to edit their penalty rules with text suggestions, enabling PenChain to generate the APIs of the needed compensation modules hosted on the provider's side.

Integration of digitized SLAs with online payment services closes the loop to full-fledged automatized service provisioning. In this study, we did not investigate the realization of payment services, as the focus of this study was on digitized SLAs. Following the Web3 vision, payment services become subject to disintermediation, compare with Section 2.3. In order to handle payments according to the Web3 paradigm, we currently investigate the integration of PenChain into the Alphabill platform that we have suggested recently [67]. Alphabill is a platform for universal asset tokenization, transfer and exchange as a global medium of exchange. Users of Alphabill can launch new so-called blockchain partitions. Each of the Alphabill partitions implements an individual token and its corresponding transaction system. The Alphabill platform provides the necessary technological fabrics that ensure governance, certification and consistency with uncapped scalability [67]. In particular, the platform has inbuilt concepts to realize multi-asset swap transactions, which are also the key to ultra-scalable decentralized payment services.

8. Conclusion

Service delivery stays in the mainstream of today's information systems. It is confronted with real challenges, e.g., higher expectations on and trust in service quality, ways to resolve disputes that may arise during a service transaction. Scholarly work in a variety of service sectors investigates novel registration and search techniques and leverages the distributed ledger technology to improve the trust of service provisioning. Innovated business and service models receive a boost from the recent advancement of the distributed ledger technologies. In this work, we look into the automatic enforcement of dynamic SLAs, which incorporate not only constraints on the service quality but also human-mediated factors such as the penalty. We identify three research questions as follows: (i) In what formal or ground logic should penalty rules be expressed? (ii) How to digitize the penalty-aware SLAs to enable an enforcement mechanism and the computation of an objective unbiased reputation in service provisioning? (iii) How to gear up a distributed ledger for enforcing the relevant penalty rules during a service transaction?

This article describes the conceptualization, design, engineering and analytical considerations of PenChain – an SLA-driven service lookup platform that extensively utilizes the blockchain technology. PenChain programmatically enforces penalty-aware SLAs and automatically assesses service providers' reputation. We argue that the blockchain-enabled enforcement mechanism in PenChain is suited to several service domains including precision agriculture. We discuss the pros and cons of PenChain with respect to service registry, service discovery and service binding. PenChain needs to be tailored or geared up for a specific service domain but the proposed service lookup and enforcement mechanism remain cross-domain.

References

[1] T. Paschou, F. Adrodegari, M. Perona, N. Saccani, Digital servitization in manufacturing: A systematic literature review and research agenda, Industrial Marketing Management 89 (2020) 278–292.

[2] A. Bouguettaya, M. Singh, M. Huhns, Q.-Z. Sheng, H. Dong, Q. Yu, A.-G. Neiat, A. Mistry, B. Benatallah, B. Medjahed, M. Ouzzani, F. Casati, X. Liu, H. Wang, D. Georgakopoulos, L. Chen, S. Nepal, Z. Malik, A. Erradi, Y. Wang, B. Blake, S. Dustdar, F. Leymann, M. Papazoglou, A Service Computing Manifesto: The Next 10 Years, Communications of the ACM 60 (4) (2017) 64–72.

[3] W. A. Awad, N. E. EL-Attar, Adaptive SLA mechanism based on fuzzy system for dynamic cloud environment, International Journal of Computers and Applications (2019) 12–22.

[4] J. M. García, O. Martín-Díaz, P. Fernandez, C. Müller, A. Ruiz-Cortés, A Flexible Billing Life Cycle for Cloud Services Using Augmented Customer Agreements, IEEE Access 9 (2021) 44374–44389.

[5] D.-T. Le, T.-V. Nguyen, L.-S. Lê, T. Kurniawan, Reinforcing Service Level Agreements in Tourism Sector – The Role of Blockchain and Mobile Computing, in: Proceedings of the 15th International Conference on Advanced Computing and Applications, IEEE, Online, 2020, pp. 160–164.

[6] T.-V. Nguyen, L.-S. Lê, H.-L. Truong, K. Nguyen-An, P. Ha, Handling Service Level Agreements in IoT= Minding Rules + Log Analytics?, in: Proceedings of the 22nd International Enterprise Distributed Object Computing Conference, IEEE Computer Society, Stockholm, Sweden, 2018, pp. 145–153.

[7] S. Tadelis, Reputation and Feedback Systems in Online Platform Markets, Annual Review of Economics 8 (1) (2016) 321–340.

[8] T. Matherly, A panel for lemons? Positivity bias, reputation systems and data quality on MTurk, European Journal of Marketing 53 (2) (2019) 195–223.

[9] G. J. Mirobi, L. Arockiam, Service level agreement in cloud computing: An overview, in: Proceedings of the International Conference on Control, Instrumentation, Communication and Computational Technologies, IEEE, Kumaracoil, India, 2015, pp. 753–758.

[10] H. Ludwig, A. Keller, A. Dan, R. P. King, R. Franck, Web service level agreement (WSLA) language specification, IBM corporation (January 2003).

[11] R. Engel, S. Rajamoni, B. Chen, H. Ludwig, A. Keller, ysla: Reusable and Configurable SLAs for Large-Scale SLA Management, in: Proceedings of the 4^{th} International Conference on Collaboration and Internet Computing, IEEE, 2018, pp. 317–325.

[12] S. Mubeen, S. A. Asadollah, A. V. Papadopoulos, M. Ashjaei, H. Pei-Breivold, M. Behnam, Management of service level agreements for cloud services in IoT: A systematic mapping study, IEEE access 6 (2017) 30184–30207.

[13] S. Yin, A. Hameurlain, F. Morvan, SLA Definition for Multi-tenant DBMS and its Impact on Query Optimization, IEEE Transactions on Knowledge and Data Engineering 30 (11) (2018) 2213–2226.

[14] W. Hussain, F. K. Hussain, O. Hussain, R. Bagia, E. Chang, Risk-based framework for SLA violation abatement from the cloud service provider's perspective , The Computer Journal 61 (9) (2018) 1306–1322.

[15] T. Labidi, A. Mtibaa, W. Gaaloul, S. Tata, F. Gargouri, Cloud SLA modeling and monitoring, in: Proceedings of the 14^{th} International Conference on Services Computing, IEEE, Honolulu, USA, 2017, pp. 338–345.

[16] V. K. Prasad, M. D. Bhavsar, Monitoring and prediction of sla for iot based cloud, Scalable Computing: Practice and Experience 21 (3) (2020) 349–358.

[17] S. Noureddine, B. Meriem, ML-SLA-IoT: an SLA Specification and Monitoring Framework for IoT applications, in: Proceedings of the 11^{th} International Conference on Information Systems and Advanced Technologies, IEEE, Tebessa, Algeria, 2021.

[18] E. J. Scheid, B. B. Rodrigues, L. Z. Granville, B. Stiller, Enabling dynamic SLA compensation using blockchain-based smart contracts, in: Proceedings of the 16^{th} IFIP/IEEE Symposium on Integrated Network and Service Management, IEEE, Washington DC, USA, 2019, pp. 53–61.

[19] M. Touloupou, E. Kapassa, C. Symvoulidis, P. Stavrianos, D. Kyriazis, An integrated SLA management framework in a 5G environment, in: Proceedings of the 22^{nd} Conference on Innovation in Clouds, Internet and Networks and Workshops, IEEE, Paris, France, 2019, pp. 233–235.

[20] A. Oluwabukola, A. Adebowale, An Architectural Model for SLA Negotiation between SaaS and Customers, International Journal of Engineering Applied Sciences and Technology 5 (5) (2020) 30–36.

[21] F. Li, G. White, S. Clarke, A Trust Model for SLA Negotiation Candidates Selection in a Dynamic IoT Environment, IEEE Transactions on Services Computing 15 (5) (2021) 2565–2578.

[22] T. Labidi, A. Mtibaa, W. Gaaloul, F. Gargouri, Cloud SLA negotiation and re-negotiation: An ontology-based context-aware approach, Concurrency and Computation: Practice and Experience 32 (15) (2020) e5315.

[23] H. Nakashima, M. Aoyama, An automation method of SLA contract of Web APIs and its platform based on blockchain concept, in: Proceedings of the 1^{st} IEEE International Conference on Cognitive Computing, IEEE, Honolulu, USA, 2017, pp. 32–39.

[24] H. Zhou, C. de Laat, Z. Zhao, Trustworthy cloud service level agreement enforcement with blockchain based smart contract, in: Proceedings of the 10^{th} IEEE International Conference on Cloud Computing Technology and Science, IEEE, Nicosia, Cyprus, 2018, pp. 255–260.

[25] A. Alzubaidi, K. Mitra, P. Patel, E. Solaiman, A blockchain-based approach for assessing compliance with SLA-guaranteed IoT services, in: Proceedings of the IEEE International Conference on Smart Internet of Things, IEEE, Beijing, China, 2020, pp. 213–220.

[26] R. Ranchal, O. Choudhury, SLAM: A Framework for SLA Management in Multicloud ecosystem using Blockchain, in: Proceedings of the IEEE Cloud Summit, Harrisburg, USA, 2020, pp. 33–38.

[27] P. Abhishek, A. Chobari, D. Narayan, SLA Violation Detection in Multi-Cloud Environment using Hyperledger Fabric Blockchain, Nitte, India, 2021, pp. 107–112.

[28] N. Neidhardt, C. Köhler, M. Nüttgens, Cloud service billing and service level agreement monitoring based on blockchain, in: Proceedings of the 9^{th} International Workshop on Enterprise Modeling and Information Systems Architectures, CEUR Workshop Proceedings, 2018, pp. 65–69.

[29] A. Wonjiga, S. Peisert, L. Rilling, C. Morin, Blockchain as a trusted component in cloud SLA verification, in: Proceedings of the 12^{th} International Conference on Utility and Cloud Computing, ACM, 2019, pp. 93–100.

[30] H. Zhou, X. Ouyang, Z. Ren, J. Su, C. de Laat, Z. Zhao, A blockchain based witness model for trustworthy cloud service level agreement enforcement, in: Proceedings of the 38^{th} IEEE Conference on Computer Communications, IEEE, Nicosia, Cyprus, 2019, pp. 1567–1575.

[31] K. Khan, J. Arshad, W. Iqbal, S. Abdullah, H. Zaib, Blockchain-enabled real-time SLA monitoring for cloud-hosted services, Cluster Computing 25 (1) (2022) 537–559.

[32] S. Battula, S. Garg, R. Naha, M. Amin, B. Kang, E. Aghasian, A blockchain-based framework for automatic SLA management in fog computing environments, Journal of Supercomputing 18 (15).

[33] M. Rahman, I. Khalil, M. Atiquzzaman, Blockchain-enabled SLA compliance for crowdsourced edge-based network function virtualization, IEEE Network 35 (5) (2021) 58–65.

[34] K. Wang, B. Yang, L. Su, Y. Hu, Blockchain based Data Sharing for User Experience Driven Slice SLA Guarantee, in: Proceedings of the 14^{th} International Conference on Service Science, IEEE, Xi'an, China, 2021, pp. 7–13.

[35] A. Kalla, C. De Alwis, P. Porambage, G. Gür, M. Liyanage, A survey on the use of blockchain for future 6G: Technical aspects, use cases, challenges and research directions, Journal of Industrial Information Integration 30 (2022) 100404.

[36] K. Xiao, Z. Geng, Y. He, G. Xu, C. Wang, W. Cheng, A blockchain based privacy-preserving cloud service level agreement auditing scheme, in: Proceedings of the 5^{th} International Conference on Wireless Algorithms, Systems, and Applications, Springer, Berlin, Heidelberg, 2020, pp. 542–554.

[37] K. Xiao, Z. Geng, Y. He, G. Xu, C. Wang, Y. Tian, A blockchain-based privacy-preserving 5G network slicing service level agreement audit scheme, Eurasip Journal on Wireless Communications and Networking (165).

[38] N. Hamdi, C. El Hog, R. Ben Djemaa, L. Sliman, A survey on SLA management using blockchain based smart contracts, in: Proceedings of the 21^{st} International Conference on Intelligent Systems Design and Applications, 2022, pp. 1425–1433.

[39] R. Uriarte, R. De Nicola, K. Kritikos, Towards distributed SLA management with smart contracts and blockchain, in: Proceedings of the 10^{th} International Conference on Cloud Computing Technology and Science, IEEE, 2018, pp. 266–271.

[40] A. Pandey, D. Narayan, K. Shivaraj, SLA Violation Detection and Compensation in Cloud Environment using Blockchain, in: Proceedings of the 12^{th} International Conference on Computing Communication and Networking Technologies, IEEE, Kharagpur, India, 2021.

[41] I. Singh, S.-W. Lee, RE-BBC: Requirements Engineering in a Blockchain-Based Cloud: Its Role in Service-Level Agreement Specification, IEEE Software 37 (5) (2020) 7–12.

[42] I. Singh, S.-W. Lee, SRE-BBC: A Self-Adaptive Security Enabled Requirements Engineering Approach for SLA Smart Contracts in Blockchain-Based Cloud Systems, Sensors 22 (10).

[43] R. B. Uriarte, H. Zhou, K. Kritikos, Z. Shi, Z. Zhao, R. D. Nicola, Distributed service-level agreement management with smart contracts and blockchain, Concurrency and Computation: Practice and Experience 33 (1425) (2021) 1–17.

[44] C. H. Liu, Q. Lin, S. Wen, Blockchain-enabled data collection and sharing for industrial IoT with deep reinforcement learning, IEEE Transactions on Industrial Informatics 15 (6) (2018) 3516–3526.

[45] M. S. Rahman, M. Chamikara, I. Khalil, A. Bouras, Blockchain-of-blockchains: An interoperable blockchain platform for ensuring IoT data integrity in smart city, Journal of Industrial Information Integration 30 (2022) 100408.

[46] F.-J. Ferrández-Pastor, J. Mora-Pascual, D. Díaz-Lajara, Agricultural traceability model based on IoT and Blockchain: Application in industrial hemp production, Journal of Industrial Information Integration 29 (2022) 100381.

[47] X. Xu, X. Zhang, H. Gao, Y. Xue, L. Qi, W. Dou, BeCome: Blockchain-enabled computation offloading for IoT in mobile edge computing, IEEE Transactions on Industrial Informatics 16 (6) (2019) 4187–4195.

[48] C.-A. Ardagna, R. Asal, E. Damiani, N.-E. Ioini, M. Elahi, C. Pahl, From trustworthy data to trustworthy IoT: A data collection methodology based on blockchain, ACM Transactions on Cyber-Physical Systems 5 (1).

[49] P. K. Sharma, J. H. Park, Blockchain based hybrid network architecture for the smart city, Future Generation Computer Systems 86 (2018) 650–655.

[50] S. Hameed, S. A. Shah, Q. S. Saeed, S. Siddiqui, I. Ali, A. Vedeshin, D. Draheim, A scalable key and trust management solution for IoT sensors using SDN and blockchain technology, IEEE Sensors Journal 21 (6) (2021) 8716–8733.

[51] S. Siddiqui, S. A. Shah, I. Ahmad, A. Aneiba, D. Draheim, S. Dustdar, Toward software-defined networking-based IoT frameworks: A systematic literature review, taxonomy, open challenges and prospects, IEEE Access 10 (2022) 70850–70901.

[52] V. Shannon, A 'more revolutionary' Web, The New York Times (May 2006).

[53] J. Wiles, What Is Web3?, Gartner – Information Technology (February 2022).

[54] M. Bennett, Web3 isn't going to fix the shortcomings of today's Web, Forrester (May 2022).

[55] B. Platz, Why Web3 is so confusing, Forbes Technology Council (June 2022).

[56] T. Stackpole, What Is Web3?, Harvard Business Review (May 2022).

[57] L. Jin, K. Parrott, Web3 Is Our Chance to Make a Better Internet, Harvard Business Review (May 2022).

[58] J. Esber, S. D. Kominers, Why build in Web3, Harvard Business Review (May 2022).

[59] T. Berners-Lee, J. Hendler, O. Lassila, The Semantic Web – A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities, Scientific American.

[60] A. Prodromou, TLS Security 2: A Brief History of SSL/TLS, Acunetix (March 2019).

[61] D. Draheim, The service-oriented metaphor deciphered, Journal of Computing Science and Engineering 4 (4) (2010) 253–275.

[62] T. Berners-Lee, Socially Aware Cloud Storage, World Wide Web Consortium (August 2009).

[63] T. Berners-Lee, Read-Write Linked Data, World Wide Web Consortium (October 2009).

[64] E. Mansour, A. V. Sambra, S. Hawke, M. Zereba, S. Capadisli, A. Ghanem, A. Aboulnaga, T. Berners-Lee, A Demonstration of the Solid Platform for Social Web Applications, in: Proceedings of the 25th International Conference Companion on World Wide Web, ACM, Montréal, Canada, 2016, pp. 223––226.

[65] D. Weinberber, How the Father of the World Wide Web Plans to Reclaim It from Facebook and Google, https://www.digitaltrends.com/ (August 2016).

[66] A. Buldas, D. Draheim, M. Gault, M. Saarepera, Towards a Foundation of Web3, in: Proceedings of the 9th International Conference on Future Data and Security Engineering, Springer, Ho Chi Minh City, Vietnam, 2022.

[67] A. Buldas, D. Draheim, M. Gault, R. Laanoja, T. Nagumo, M. Saarepera, S. A. Shah, J. Simm, J. Steiner, T. Tammet, A. Truu, An ultra-scalable blockchain platform for universal asset tokenization: Design and implementation, IEEE Access 10 (2022) 77284–77322.

[68] C. Lagarde, Central Banking and Fintech – A Brave New World?, Bank of England Conference, London (September 2017).

[69] A. Buldas, D. Draheim, T. Nagumo, A. Vedeshin, Blockchain technology: Intrinsic technological and socio-economic barriers, in: Proceedings of the 7^{th} International Conference on Future Data and Security Engineering, Springer, Quy Nhon, Vietnam, 2020, pp. 3–27.

[70] J. Fernández-Villaverde, D. Sanches, L. Schilling, H. Uhlig, Central bank digital currency: Central banking for all?, Review of Economic Dynamics 41 (2021) 225–242.

[71] L. Grassi, D. Lanfranchi, A. Faes, F. M. Renga, Do we still need financial intermediation? The case of decentralized finance – DeFi, Qualitative Research in Accounting & Management 19 (3) (2022) 323–347.

[72] D. A. Zetzsche, D. W. Arner, R. P. Buckley, Decentralized Finance, Journal of Financial Regulation 6 (2) (2020) 172–203.

[73] Y. Chen, C. Bellavitis, Blockchain Disruption and Decentralized Finance: The Rise of Decentralized Business Models, Journal of Business Venturing Insights 13 (e00151).

[74] F. Schär, Decentralized finance: On blockchain- and smart contract-based financial markets, Federal Reserve Bank of St. Louis Review 103 (2) (2021) 153–74.

[75] International Organization for Standardization, ISO 29003:2018 – Information technology — Security techniques — Identity proofing, ISO, 2018.

[76] S. Lips, N. Bharosa, D. Draheim, eIDAS Implementation Challenges: the Case of Estonia and the Netherlands, in: Proceedings of the 7^{th} International Conference on Electronic Governance and Open Society: Challenges in Eurasia, Springer, St. Petersburg, Russia, 2020, pp. 75–89.

[77] A. Mühle, A. Grüner, T. Gayvoronskaya, C. Meinel, A survey on essential components of a self-sovereign identity, Computer Science Review 30 (2018) 80–86.

[78] A. Oram, Peer to Peer: Harnessing the Power of Disruptive Technologies, O'Reilly, 2001.

[79] A. Andjelic, How Brands are Experimenting with Web3, Harvard Business Review (May 2022).

[80] S. Wang, W. Ding, J. Li, Y. Yuan, L. Ouyang, F.-Y. Wang, Decentralized Autonomous Organizations: Concept, Model, and Applications, IEEE Trans. Computat. Social Syst. 6 (5) (2019) 870–878.

[81] M. Niforos, The promising future of NFTs remains in a state of flux, Financial Times (June 2022).

[82] M. Dowling, Fertile LAND: Pricing non-fungible tokens, Finance Research Letters 44 (102096) (2022) 2–5.

[83] R. Sharma, What Is a Non-Fungible Token (NFT)?, Investopedia (January 2023).

[84] P. F. Linington, Z. Milosevic, J. Cole, S. Gibson, S. Kulkarni, S. Neal, A Unified Behavioural Model and a Contract Language for Extended Enterprise, Data & Knowledge Engineering 51 (1) (2004) 5–29.

[85] D.-M. Gabbay, J. Woods, Logic and the Modalities in the Twentieth Century, Volume 7 (Handbook of the History of Logic), North Holland, 2006.

[86] S. Bistarelli, Semirings for Soft Constraint Solving and Programming, Springer, 2004.

[87] W. Kettinger, C. Lee, Perceived Service Quality and User Satisfaction with the Information Services Function, Decision sciences 25 (5-6) (1994) 737–766.

[88] R. Saket, N. Singh, P. Dayama, V. Pandit, Smart Contract Protocol for Authenticity and Compliance with Anonymity on Hyperledger Fabric, in: Proceedings of the 4th International Conference on Blockchain and Cryptocurrency, IEEE, Virtual Conference, 2020, pp. 1–9.