Journal of Universal Computer Science, vol. 29, no. 7 (2023), 691-717
submitted: 3/11/2022, accepted: 13/3/2023, appeared: 28/7/2023 CC BY-ND 4.0

Evaluation of a Legally Binding Smart-Contract Language for Blockchain Applications

Vimal Dwivedi
(Queen's University Belfast, Northern Ireland, UK
https://orcid.org/0000-0001-9177-8341, v.dwivedi@qub.ac.uk)

Mubashar Iqbal
(University of Tartu, Tartu, Estonia
https://orcid.org/0000-0003-0543-613X, mubashar.iqbal@ut.ee)

Alex Norta
(Tallinn University, Tallinn, Estonia
https://orcid.org/0000-0003-0593-8244, alex.norta.phd@ieee.org)

Raimundas Matulevičius
(University of Tartu, Tartu, Estonia
https://orcid.org/0000-0002-1829-4794, rma@ut.ee)

Abstract: Blockchain governs inter-organizational business processes and enables decentralized autonomous organizations (DAO) with governance capabilities via smart contracts (SC). Due to the programmer's lack of prior knowledge of the contract domain, SCs are ambiguous and error-prone. Several works, i.e., SPESC, Symboleo, and SmaCoNat, exist to support the legally-binding SCs. The aforementioned SCLs present intriguing approaches to building legally-binding SCs but either lack domain completeness, or are intended for non-collaborative business processes. In our previous work, we address the above-mentioned shortcomings of the XML-based smart-legal-contract markup language (SLCML), in which blockchain developers focus on the contractual workflow rather than the syntax specifics. However, SLCML, as a blockchain-independent formal specification language, is not evaluated to determine its applicability, usefulness, and usability for establishing legally-binding SCs for workflow enactment services (WES) to automate and streamline the business processes within connected organizations. In accordance with this, we formally implement the SLCML and propose evaluation approaches, such as running case and lab experiments, to demonstrate the SLCML's generality and applicability for developing legally-binding SCs. Overall, the results of this work ascertain the applicability, usefulness, and usability of the proposed SLCML for establishing legally-binding SCs for WES.

Keywords: Blockchain, Legally-binding smart contract, Decentralized autonomous organization, Smart contract language.
Categories: C.2.0, C.2.4, D.1.3, D.3.2
DOI: 10.3897/jucs.97112

1 Introduction

Many businesses use workflow enactment services (WES) to automate and streamline their business processes to enhance efficiency, reduce cost, and increase profits [Pourmirza et al., 2019]. Although each organization is unique, many work processes are common across all organizations within a specific industry. As a result, the WESs are designed in such a way that they can be configured to meet the needs of a particular organization. In this way, the WESs are equipped to handle common business processes in many different types of organizations and across industries [Yussupov et al., 2022]. When deciding on WES, most organizations search for a competent workflow vendor who can satisfy their requirements. Collaboration between individuals and organizations requires communication and cooperation (e.g., collaboration involves multiple business processes). Each organization has its procedures and workflows, and effective collaboration requires that the WES of each collaborating organization can communicate and cooperate. Information technology (IT) professionals have long struggled to preserve consistency and mutual trust in inter-organizational business processes [Abodei et al., 2019a], [Kormiltsyn et al., 2019]. Information on business operations can be shared and validated within an organization's centralized business processes where participants trust one another. When control over a process is delegated outside of an organization, as in an inter-organizational collaboration (IOC) process, either organization cannot validate data accuracy, enforce contractual obligations, or ensure that specific conditions are met. Consequently, transferring control between fragile business processes across organizations can lead to inconsistency and a lack of trust in process management [Matulevičius et al., 2017].

Blockchain has the potential to execute, monitor, and improve business processes within an organization [Mendling et al., 2018]. Blockchain is a distributed ledger technology based on a timestamped list of transactions that cannot be tampered [Iqbal and Matulevičius, 2021]. Blockchain provides the execution of business processes in a decentralized environment, aided by the peer-to-peer (P2P) network, consensus, cryptography, and immutable ledger. The secure nature of blockchain plays an important role in the development and implementation of smart contracts (SC) that are self-executing computer programs designed to automate business operations [Lin et al., 2018]. Established SC languages (SCL), such as Solidity or Vyper, are high-level programming languages for writing SCs. Unfortunately, due to their technical and specialized nature, the SCs developed using these languages are not understood by non-IT practitioners [Dwivedi et al., 2021b]. Even for IT specialists, assessing the legal requirements of SCs is challenging and time-consuming due to their lack of legal knowledge [He et al., 2018].

To address the aforementioned gaps, our previous work [Dwivedi et al., 2021a] proposes a blockchain-independent formal specification language, i.e., SLCML, that allows blockchain developers to focus on the contractual workflow rather than the blockchain-specific programming code. However, such a blockchain-independent formal specification language is not evaluated to determine its applicability, usefulness, and usability for establishing legally-binding SCs for WES to automate and streamline the business processes within connected organizations. Thus, the main goal of this study is to implement the legally-binding SCL described in [Dwivedi et al., 2021a], then perform the developed SCL evaluation. The syntax of the proposed language has previously been evaluated for correctness and consistency in [Dwivedi et al., 2021a]. In contrast, we aim to determine the applicability, usefulness, and usability of the proposed SCL in describing legally binding SCs in this work, which is based on my doctoral thesis [Dwivedi, 2022]. Following this, we propose the main research question *how to evaluate legally-binding SCs language?* To establish a separation of concerns, we deduce the following subquestions: What is the machine-readable language conversion for legally-binding SCs? What is the structure of the proposed language instantiation? What is the usefulness of the proposed language in drafting legally-binding SCs? The main contributions of this paper are as follows.

1. We formally implement the legally-binding SCL using the blockchain-independent

formal specification language discussed in [Dwivedi et al., 2021a].

2. We propose novel evaluation approaches to evaluate the legally-binding SCL for blockchain-based applications.
3. We perceive the usefulness and usability of the proposed SCL in describing the legally-binding SCs.

The aforementioned contributions deliver a theoretical contribution. Similarly, the practical contribution is essential for scientific understanding of a given topic of interest and states phenomena as they are [Zhou et al., 2017]. The practical contribution relates to how the suggested evaluation approaches are used and put into practice. We provide a use case as an illustration to explain the acceptance and actual use of legally-binding SCs. The thorough analysis offered by the SLCML instantiation, semantic, pragmatic, and usability assessments bring another practical contribution to this study. This practical contribution, for instance, seeks to comprehend, based on theoretical presumptions, how the proposed SCL may be institutionalized and how it helps create legally-binding SCs based on blockchain-independent formal specification language.

The development of artifacts in this paper adheres to the design-science research (DSR). The DSR provides a rigorous framework for the creation and evaluation of the designed artifacts and to solve practical problems [Von Alan et al., 2004]. We follow the six steps of the DSR, including: *(i)* identifying a problem (e.g., evaluation of the legal and conceptual business problem in SCLs), *(ii)* defining requirements (e.g., identifying the most appropriate evaluation approaches), *(iii)* developing artifacts (e.g., development of robust and ease of use smart-contract language for blockchain community), *(iv)* evaluation of artifacts (e.g., identifying and conducting experiments in a real-time environment to evaluate SCL), *(v)* improving artifacts (e.g., conducting webinar and workshop to improve developed SCL), *(vi)* results communication (e.g., providing SCL to the scientific community to develop SCs for blockchain-based applications).

The remainder of the paper is structured as follows: Section 2 discusses the background and related work. Section 3 presents the smart legal contract markup language (SLCML) for drafting legally-binding SCs. Section 4 discusses examples of SC code using the SLCML schema, evaluation approaches, and evaluation results. Section 5 presents the discussion and future work. Section 6 concludes the paper.

2 Background and Related Work

This section introduces the fundamental concepts of blockchain before demonstrating the legal implications of blockchain SCs. Section 2.1 provides a technical overview of SCLs, after which the legal implications of SCs are explained in more detail. Section 2.2 discusses the state of the art of smart contract development. Section 2.3 compares the related work and expresses the motivation for this paper.

2.1 Blockchain and Smart Contracts

The emergence of blockchain technology can be traced back to the inception of the Bitcoin cryptocurrency [Nakamoto, 2008]. In this first use case, a blockchain's decentralized nature enables the transfer of cryptocurrency without the involvement of a central banking or financial authority and thereby eliminates the cost of bank fees, taxes, and other intermediary expenses during transactions. A blockchain is an immutable distributed ledger to store data in a variety of domains such as business, healthcare, passport verification, and so on [Abodei et al., 2019b]. Note that blockchains are not intended for mass-data storage. Bitcoin uses SHA-256 hashing cryptography algorithms for the security of storing cryptocurrency transaction logs. Blockchain has significant potential in internet-of-things (IoT) applications because it strengthens device security and data obscurity, while also improving maintainability. The main technologies that support blockchains include decentralized consensus and storage, public-key cryptography and asymmetric encryption, SCs, and smart contracting languages.

***Decentralized consensus**:* The process of updating records in a blockchain network to ensure that new information is accurate and consistent is known as blockchain consensus [Swan, 2015]. A blockchain consensus ensures that only correct data, validated by collaborating parties, is added to the network for an IOC in a decentralized environment. As a result of blockchain consensus, transparency in IOC processes is improved. Proof-of-work (PoW) is the primary consensus mechanism used by Bitcoin and Ethereum cryptocurrencies. Bitcoin is the fastest-growing blockchain network, and Ethereum is the largest blockchain network for executing SCs [Bartoletti and Pompianu, 2017]. In the PoW consensus mechanism, a complex mathematical puzzle is presented, and the network's first member to solve it is chosen to add the next record to a blockchain ledger. Because of the scalability and resource consumption issues with PoW, Jain et al. investigated a proof-of-stake (PoS) consensus method as a better solution. Participants in a PoS network are selected to add the next record to the ledger based on the amount of stake they have deposited [Jain et al., 2018].

***Public-key cryptography (PKC)**:* The PKC is a system that uses a public-and-private key pair to identify participants in a decentralized network uniquely. The public key is used to identify IOC participants, while the private key is used to sign transactions. As a result, PKC provides a tamper-proof source verification system for any activity in an IOC. The asymmetric encryption provided by the PKC can be used to implement access control on IOC processes, thus ensuring that specific IOC functions can only be performed by certain network parties. As a result, PKC significantly addresses the security issues currently encountered in traditional blockchain systems for executing SCs within IOCs that use decentralized collaboration.

***Decentralized storage**:* A blockchain network's records are replicated in all participating nodes [Pilkington, 2016]. Decentralized databases can be used to extend existing blockchain storage by providing additional repositories for blockchain systems [Croman et al., 2016], thus preserving digital assets exchanged in IOC-executed blockchain networks. As a result, for an IOC that employs decentralized collaboration concepts, decentralized storage ensures that data resulting from the execution of IOC functions are accessible in real-time to all participants. As a result, interoperability issues in current IOC systems are eliminated by real-time data access.

***Smart contracts**:* Smart contracts are blockchain-based applications that allow businesses to be more efficient by automating business processes [López-Pintado et al., 2019]. A smart contract is a computerized transaction that enforces agreement rules automatically without the use of intermediaries [Szabo, 1997]. Organization collaboration processes can be reconstructed into SC workflows and executed on blockchain networks. Smart contracts are programmable with business logic and rules, ensuring that IOCs' SCs are executed securely without requiring centralized authority. Blockchain-enabled smart contracts do not interact with external data; they rely solely on data provided by the blockchain system to carry out business logic. Decentralized oracles are external data-gathering components in a blockchain that allow SCs to receive real-world data inputs without relying on centralized parties [George and Lesaege, 2020]. This ensures that SCs use trusted data inputs when executing business logic in IOCs.

***Smart contract language**:* Smart contracts are written in a programming language, such as Solidity, with intermediate languages, e.g., Simplicity [Valliappan et al., 2018] and Scilla [Sergey et al., 2019] for program analysis and verification. The latter provides significant assurances by relying on type-soundness. These languages allow SC code to run on low-level virtual machines (VM) (Ethereum VM, for example). Rootstock [1], Telegram Open Network (TON) [2], and Bitcoin [Khalil et al., 2017] are just a few of the blockchains that, just as Ethereum, have designed and implemented, their virtual machines. Rootstock, for example, comprises the Rootstock Virtual Machine (RVM) to Bitcoin [Khalil et al., 2017], whereas TON introduced the TON VM, or TVM, for developing, maintaining, and configuring SCs[Durov, 2019].

Legal implications of smart contracts: For contracts to be legally-binding, the parties to the contract must reach an agreement. According to Governatori et al., [Governatori et al., 2018], the conceptual links between legal, commercial contracts, and smart contracts are that SCs must meet specific requirements in order to be legal contracts. These requirements include offer and acceptance, consideration, competence, capacity to contract, and so on. According to Savelyev [Savelyev, 2017], SCs are in accordance with Roman contract law he explains this by comparing smart contracts to the mechanism of a vending machine [Dwivedi and Norta, 2021]. When using a vending machine, an individual places a coin in the slot that leads to a secure lockbox. The individual then selects a specific product from a predefined list, and if the money deposited in the machine is of the same value as the product selected, the vending machine automatically dispenses the product.

Savelyev also proposes blockchain decentralization as a solution for aligning with government power. The proposed solutions are based on granting state authorities the ability to modify blockchain databases as superusers while emphasizing traditional remedies and enforcement practices. Furthermore, Goldenfein et al., [Goldenfein and Leiter, 2018] argue that the legal status of SCs is dependent on incorporating computational transactions into natural contracts because "natural contracts do not construct an agreement on their own". De Filippi et al., [Filippi and Hassan, 2016] define a smart contract as "law is code" and propose abandoning the concept of "code is the law." It is worth noting that when contract law is translated into SC code [Farrell et al., 2017], the semantics of contract law are lost, thereby leaving the legal status of SCs in doubt.

2.2 State of the Art of Smart Contract Development

SCs using distributed ledger technology is the latest research and little research has been conducted on SCLs supporting the development of legally-binding SCs based on the requirements and properties of actual business processes. To support these contractual properties, attempts have been made to raise the level of abstraction from code-centric to model-centric SC development. This section describes four of these attempts in detail: the agent-based approach, the business process-based approach, the state machine approach, and the UML approach. These model-driven approaches have used one or more modeling languages to support the concerns and viewpoints associated with SCs.

Agent-based approach: Frantz et al., [Frantz and Nowostawski, 2016] developed a modeling approach that used a domain-specific language (DSL) to help transform

[1] Rootstock (RSK) | Home Page https://www.rsk.co/
[2] TON | GitHub Page https://github.com/ton-blockchain/ton

institutional concepts into machine-readable contractual rules. This approach was based on the concepts discussed in [Crawford and Ostrom, 1995]. Crawford & Ostrom proposed a 'grammar of institutions' syntax that can be used to identify essential components of any institution and group them into three types of institutional statements: rules, norms, and shared strategies. These institutional statements are useful in agent-based modeling because they serve to guide the actions of agents within organizations [Smajgl et al., 2010]. Agent-based models (ABMs) are computational models for simulating the actions and interactions of individual agents, or groups of agents, within a system, for understanding agent behavior and the system's behavior as a whole. This includes the interactions of the system's entities as well as specific representations of those entities [Shekhar and Xiong, 2008]. The syntax of the grammar of institutions contains five components represented by the acronym ADICO, which stands for: Attributes, Deontic, aIm, Conditions, and Or Else. Frantz et al., [Frantz and Nowostawski, 2016] developed a DSL in Scala to convert the ADICO statements into Solidity code. In this way, a business contract written in ADICO syntax could easily be converted into a SC using Solidity. Thus, the generated SCs would be understood by both IT, and business professionals. Still, the code generated through this process is merely a skeleton and requires enhancement by a developer before running in Solidity. Furthermore, no mention is made in this paper to create complex collaborative business contracts.

Business process-based approach: This approach focuses on organizational business processes, and there have been several initiatives based on this approach published in the literature. To address the lack of trust in collaborative process execution in the blockchain, Weber et al., [Weber et al., 2016] proposed an automated way to generate smart contracts using a translator. This is achieved by generating SCs from process specifications using Business Process Model and Notation (BPMN). Because the translator is called at design time and the roles of the participants are unknown, the output of this process results in what is known as a factory, or generic, contract. As a result of the transformation's adherence to workflow patterns, not all BPMN elements are capable of translation. Taking a similar approach, Tran et al., created Lorikeet; a model-driven engineering tool that generates SCs from BPMN specifications [Tran et al., 2018]. The modeler user interface is linked to three back-end components: a BPMN translator, a registry generator, and a blockchain trigger. A business process model is fed into the BPMN translator, which generates a smart contract written in Solidity code. The trigger communicates with an Ethereum blockchain node to compile, deploy, and interact with SCs. Unfortunately, Lorikeet does not support all BPMN notations for translation, similar to the translator proposed in a previous study [Weber et al., 2016]. Caterpillar [Orlenyslp, 2019] is a tool available in the market that is an alternative to Lorikeet. It is a free, open-source tool that supports advanced BPMN control flow elements such as sub-processes, multiple instances, and event handling. Still, Caterpillar does not support the modeling of business-process views, which is essential for any collaborative business contracts.

State machine approach: Several studies have used the state machine approach to extend model-driven engineering concepts to smart contract development [Dixit et al., 2022]. This researcher considers SCs to be state machines; the contract has an initial state, which changes as transactions are completed. This method is a common pattern in Solidity documentation [Dannen, 2017]. Mavridou et al., [Mavridou and Laszka, 2017] presented a formal model for modeling SCs and created the FSolidM tool, a code generator for creating Ethereum smart contracts, thus enabling smart-contract development with minimal manual coding. In addition, Mavridou et al., demonstrated the FSolidM tool in [Mavridou and Laszka, 2018]. Although the transformation from finite state machine (FSM) to Solidity is semi-automated to ensure good code quality, several other properties cannot be modeled without using additional plugins. Mavridou et al., [Mavridou and Laszka, 2018] did not go into detail about FSM; they expanded on previous work by developing the 'VeriSolid' framework, which focused on the security aspect of smart contract design [Mavridou et al., 2019]. The VeriSolid framework enabled developers to perform high-level verification of smart contracts, thus allowing for correct-by-design contract development. The tool 'Yakindu Statechart Tool' is designed to generate SC code from a finite state model. The tool allows for graphical editing of state charts and code generation in blockchain languages, including Solidity, Vyper, and Yul [Mülder, 2019]. This tool is still in its initial development phases, and no details on how it can be implemented are available.

UML approach: Syahputra et al. [Syahputra and Weigand, 2019] generate SCs for two different blockchains using UML and OCL. The smart-contract code stems from an existing code generator called 'Acceleo' (Model to Text) [Home, 2005]. Kruijff et al., [de Kruijff and Weigand, 2017] employ the commitment-based ontology perspective that segregated the ontology into three levels: essential, infological, and data logical. Still, this study [de Kruijff and Weigand, 2017] lacks detail by only demonstrating the models without discussing platform-specific models or generated code. It is also unclear whether the author's framework integrates the two tools into a single system or whether developers require other tools to achieve their goal.

2.3 Related Work

In this section, we discuss the existing research on evaluating modeling languages, frameworks, and support tools for model implementation. Their benefits and drawbacks are identified and weighed to determine the suitable method for evaluating the SLCML. For example, Table 1 lists various research approaches for assessing modeling languages and support tools. The *Study* column shows the article being reviewed, and the *Domain* column shows the domain to which the modeling concept is applied. The elements used to implement the modeling language are described in the *Notation* column. Finally, the *Evaluation* column tracks the assessed modeling-language aspects.

The first article [Brandtner and Helfert, 2018] describes a systematic technique for evaluating a modeling language's syntax, semantics, and usefulness. The developed evaluation technique is applied to assess the qualitative aspects of a modeling language in organizational innovation management. The second article [Mahunnah et al., 2018] presents a quantitative method for evaluating a support tool for an agent-modeling language used in software engineering. The third article[Halvorsrud et al., 2016] develops a modeling language for customer-journey mappings to evaluate the support tool by measuring the correctness and utility of the models generated. The fourth article [Jaccheri and Stålhane, 2001] compares the results obtained using the support tool to the results obtained using the standard tool to evaluate a new modeling technique for software engineering. The fifth article [Morandini et al., 2011] evaluates the significance of the methods proposed by assessing the semantic characteristics of a modeling language that extended typical agent-oriented software-engineering methodologies. The sixth article [Opdahl and Henderson-Sellers, 2002] provides a systematic method for analyzing UML semantic features. Finally, the study [Dranidis, 2007] develops a UML support tool and evaluates the consistency of models created with it, as well as the tool's utility compared to traditional modeling techniques.

The key finding (Table 1) shows that most studies compare the utility of a modeling language's support tool to that of a traditional modeling language. Only one study [Brandtner and Helfert, 2018], provides a comprehensive explanation of the evaluations, such as syntactic and semantic correctness and modeling language's general applicability to the domain. Although the objective of the usability evaluation of the artifacts conducted in study [Mahunnah et al., 2018] is similar to that of other studies, Mahunnah et al.,'s study is more relevant because the notations evaluated are similar to this paper. The SLCML is a modeling language used to create SCs for deployment on a blockchain, and as a result, both studies, [Brandtner and Helfert, 2018] and [Mahunnah et al., 2018], are preferred as this paper's evaluation method. These two approaches were selected for evaluating the proposed SLCML because they evaluate the semantic, pragmatic, and usability aspects of the modeling language, which is an objective of this study.

Study	Title	Notation	Evaluation		
			Pragmatic	Semantic	Usability
[Brandtner and Helfert, 2018]	Multi-media and web-based Evaluation of Design artefacts-Syntactic, Semantic and Pragmatic Quality of Process Models	BPMN	++	++	++
[Mahunnah et al., 2018]	An empirical evaluation of the requirements engineering tool for socio-technical systems	AOM	+	-	+
[Halvorsrud et al., 2016]	Evaluation of modeling language for customer journeys	CJML	+	-	+
[Jaccheri and Stålhane, 2001]	Evaluation of the E3 Process Modeling Language Tool for the Purpose of Model creation	E3 Process	-	-	+
[Morandini et al., 2011]	Empirical Evaluation of Tropos4AS Modeling	Tropos-	-	+	-
[Opdahl and Henderson-Sellers, 2002]	Ontological evaluation of the UML using the Bunge-Wand-Weber model	UML	-	++	-
[Dranidis, 2007]	Evaluation of StudentUML: an Educational Tool for Consistent Modeling with UML	UML	+	-	+

Table 1: Evaluation methods for modeling languages and support tools assessment

3 Formal Specification Language

In this section, we describe the elements of the SLCML that are developed based on the legally-smart contract ontology 2.0 discussed in [Dwivedi et al., 2021a]. The extended proposed SCL ontology (legally-smart contract ontology 2.0) shown in figure 1 incorporates the legal concepts and properties for contractual collaboration in business DAOs. Since the ontology is beyond the scope of this paper, we refer the reader to [Dwivedi et al., 2021a] for an ontology description. The ontology is translated into a machine-readable language in this paper as SLCML. For the SC to be legally-binding, it must be created using a programming language that contains necessary legal elements. To address this issue, we collaborated with a lawyer to improve the existing eSourcing ontology by adding legally-relevant concepts to eSourcing ontology 1.0 [Norta, 2015].

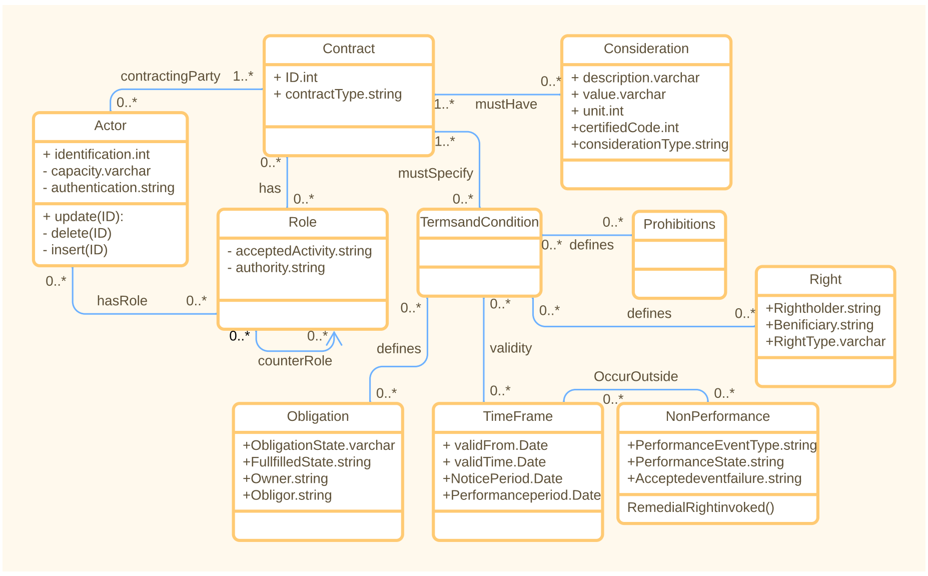

Figure 1: Upper-level smart-contract ontology, adapted from [Dwivedi et al., 2021a]

Legal-smart-contract ontology 2.0 we develop by mapping a legally relevant ontology against the eSourcing ontology 1.0 [Norta, 2015] that is the foundation for eSML 1.0, an eSouring Markup Language designed to provide answers to three critical contractual questions: Who? Where? and What? The "Who-concept" describes the parties engaged in the contracting process. The "Where-concepts" distinguish the fundamental aspects of the context of an electronic-contract, and "What-concepts" define the exchanged values and their associated conditions. The primary goal of eSML 1.0 was to enable SC collaboration within the eSourcing domain. We use Liquid Studio[3] to translate the concepts and properties of the SCL ontology into the eSML language, which is an XML schema editor for creating XML documents. This translation resulted in the creation of eSML 2.0. The SLCML is an expanded version of eSML 2.0. As this study is only

[3] Liquid Studio | Home https://www.liquid-technologies.com/xml-studio

concerned with the expanded version of SLCML, a link is provided to the complete SLCML schema [4]. The SLCML schema of the upper-level SC is presented in Section 3, and Section 3 the schema for defining domain-specific contractual properties.

Upper-level SC: The legal elements described in the upper layer of legally relevant SC DAOs are defined in the code extract (Listing 1). Line 5 of Listing 1 defines the contractual considerations and variable types. The value of minOccurs and maxOccurs in Line 5, represents the amount of consideration required for a legally-binding smart contract. Line 6 defines the terms_and_conditions element that specifies the smart contract's terms and conditions. The contracting party's description is defined in Line 7 of Listing 1, followed by the custom type company_info that includes the contracting party's name, type of legal organization, and company contact information.

```
1  <xs:element name=”contract”>
2      <xs:complexType>
3          <xs:sequence>
4              <xs:element name=”role” type=”
       variables_def_section” minOccurs=”0” maxOccurs=”unbounded
       ”/>
5              <xs:element name=”consideration” type=”
       variables_def_section” minOccurs=”1” maxOccurs=”unbounded
       ”/>
6              <xs:element name=”terms_and_conditions” type=”
       terms_and_condition_definition” minOccurs=”0” maxOccurs=”
       unbounded”/>
7              <xs:element name=”party” type=”company_info”
       maxOccurs=”unbounded” />
8              <xs:element name=”mediator” type=”company_info”
       minOccurs=”0” maxOccurs=”unbounded” />
9          </xs:sequence>
10             <xs:attribute name=”contract_id” type=”xs:ID” />
11             <xs:attribute name=”global_language” type=”xs:
       string” />
12             <xs:attribute name=”web_service_uri” type=”xs:
       string” />
13     </xs:complexType>
14 </xs:element>
```

Listing 1: Upper layer of smart contract schema

Listing 2 displays the rights, prohibitions, obligations, and time-frames defined by the custom-variable: *terms_and conditions_definition_type*. The code extract in Listing 2 is part of the terms and conditions that define the rules and regulations governing the parties' performance, as discussed in[Dwivedi et al., 2021a]. Line 3 defines the rights of the elements, as well as the custom type, (the right__type), which allows the parties to customize the type of rights. minOccurs and maxOccurs indicate that parties must choose at least one right. Prohibitions and definitions of the prohibitions that may apply to the terms and conditions are described in Line 4. Line 5 of Listing 2 specifies the obligations as well as the obligation__category that allows the parties to configure multiple obligations. Finally, the time__frame is defined in Line 6, which indicates when the terms and conditions expire.

```
1  <xs:complexType name=”terms_and_conditions_definition”>
2      <xs:sequence>
```

[4] shorturl.at/uBHR6

```
        <xs:element name=”right” type=”right_type” minOccurs=
    ”1” maxOccurs=”unbounded” />
        <xs:element name=”prohibitions” type=”xs:string”
    minOccurs=”0” />
        <xs:element name=”obligation” type=”
    obligation_category” minOccurs=”1” maxOccurs=”unbounded”
    />
        <xs:element name=”time_frame” type=”
    variables_def_section” minOccurs=”0” />
    </xs:sequence>
</xs:complexType>
```

Listing 2: Schema definition of terms and conditions

The variables_def_section, is a common variable attribute defined in Listing 3 that contains properties for all SLCML variables, both simple and complex. The string data items necessitate the definition of the string_type. The role of the contracting party, for example, could be specified as a string_type. The boolean data type is required to define boolean contract data items. For example, the boolean data type determines whether or not the contract is legally-binding. The integer data type is used to store contract-id and consideration values. Special data types, such as money_type and event_type, define specific contractual activities. For example, the money_type specifies the amount of money in a specific currency, whereas the event_type specifies the type of event that may occur during the contract.

```
<xs:complexType name=”variables_def_section”>
    <xs:sequence maxOccurs=”unbounded”>
       <xs:choice>
           <xs:element name=”string_var” type=”string_type”
    />
           <xs:element name=”real_var” type=”real_type” />
           <xs:element name=”integer_var” type=”integer_type”
     />
           <xs:element name=”boolean_var” type=”boolean_type”
     />
           <xs:element name=”date_var” type=”date_type” />
           <xs:element name=”time_var” type=”time_type” />
           <xs:element name=”event_var” type=”event_type” />
           <xs:element name=”money_var” type=”money_type” />
           <xs:element name=”external_resource_reference_var”
     type=”external_resource_reference_type” />
           <xs:element name=”list_of_events_var” type=”
    list_of_events_type” />
           <xs:element name=”list_of_strings_var” type=”
    list_of_strings_type” />
           <xs:any namespace=”targetNamespace” />
     </xs:sequence>
 </xs:complexType>
```

Listing 3: Common variable attributes

Obligation-type: The obligation_category consists of the obligation_type, obligation_state, performance and non-performance specified in Listing 4. The element obligation_type, along with custom variable obligation_type_definition, is specified in Line 3; by which several obligations are configured. The obligation_state is defined in Line 4 to monitor the contract fulfillment through which an obligation can pass.

In Listing 4, the definition of obligation__type__def-inition is omitted. The obligation state depends on the performance and non-performance conditions defined in Line 5.

```
1 <xs:complexType name=”obligation_category”>
2     <xs:sequence>
3         <xs:element name=”obligation_type” type=”
      obligation_type_definition” minOccurs=”1”/>
4         <xs:element name=”obligation_state” type=”
      obligation_state_definition” minOccurs=”1”/>
5         <xs:element name=”performance” type=”
      variables_def_section” minOccurs=”1” maxOccurs=”unbounded
      ”/>
6         <xs:element name=”non−performance” type=”
      variables_def_section” minOccurs=”0” maxOccurs=”unbounded
      ”/>
7     </xs:sequence>
8 </xs:complexType>
```

Listing 4: Schema of obligations category

Listing 5 is an example of an obligation type from which the parties can create at least one, and possibly more, obligations. The legal obligation is defined on Line 3, along with the string variable type. Business obligations have both monetary and non-monetary implications for which monetary and non-monetary elements are defined in Lines 4 and 5. Line 6, on the other hand, specifies both the string type and the moral obligation. The remaining obligations are defined similarly, as shown in Listing 5.

```
1 <xs:complexType name=”obligation_type_definition”>
2   <xs:sequence>
3     <xs:element name=”legal” type=”xs:string” minOccurs=”0”
      />
4     <xs:element name=”monetary” type=”xs:string” minOccurs=”0
      ” />
5     <xs:element name=”non−monetary” type=”xs:string”
      minOccurs=”0” />
6     <xs:element name=”moral” type=”xs:string” minOccurs=”0”
      />
7     <xs:element name=”Primary” type=”xs:string” minOccurs=”0”
       />
8     <xs:element name=”Secondary” type=”xs:string” minOccurs=”
      0” />
9     <xs:element name=”Conditional” type=”xs:string” minOccurs
      =”0” />
10    <xs:element name=”reciprocal” type=”xs:string” minOccurs=
      ”0” />
11    <xs:element name=”reconciliatory” type=”
      business_event_types” minOccurs=”0” />
12  </xs:sequence>
13 </xs:complexType>
```

Listing 5: Schema of the type of obligation

4 Evaluation of SLCML

We provide a systematic evaluation of proposed SLCML to demonstrate their applicability in developing legally-binding SCs. The generality and applicability of the proposed language are tested in subsection 4.2 by drafting the dairy supply chain contract (explained in Section 4.1) using the proposed SLCML language.

We held a webinar event with blockchain experts to review the concepts and showcase the SLCML schema. The experts work in the supply chain, healthcare, finance, education, and research industries. A total of 20 people registered for the webinar. Figure 2 provides detailed information on the background of domain experts that evaluated the SLCML. The figure shows the industry background, job positions, domain field, and experience level of the participants. About 37% of the participants are from research and education backgrounds, 25% are from supply chain and fintech, and 13% are from healthcare. Most of the participants are highly experienced, as 80% have an experience level of 2 years and above. Regarding the field domain, 60% of the participants are from academics, while 40% are from the industry. All of the important SLCML schema concepts, such as business and legal aspects and relationships, were discussed during the workshop. In addition, the attendees were introduced to the concept of decentralized inter-organizational business collaboration, the main theme of this paper. The participants were introduced to the dairy supply chain use case and asked to write the XML code using the proposed SLCML language. Thereafter, the participants were asked to provide feedback on the SLCML schema's semantic, pragmatic, and usability features at the end of the webinar.

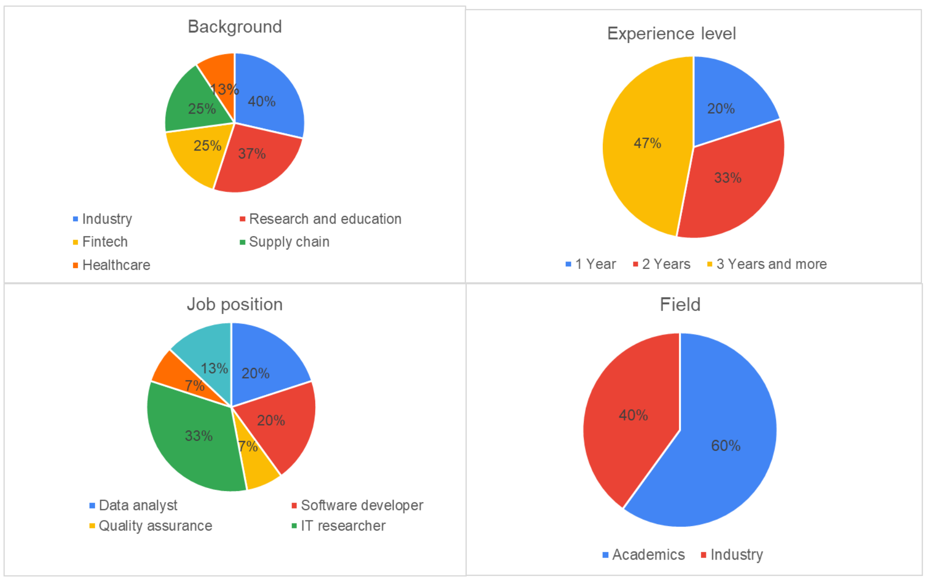

Figure 2: Demographics of domain experts

4.1 Use Case

This section discusses the ongoing case study from the dairy food supply chain for the SLCML evaluation. In lab experiments, participants write SLCML code corresponding to the case study. The case elaborated in [Behnke and Janssen, 2020] and published in [Dwivedi and Norta, 2022], describes numerous problems that can arise when a service provider hides process details from the service consumer in the dairy supply chain. Blockchain may benefit the food supply chain, such as the fish supply chain [Howson, 2020], and the dairy supply chain. The tracking and monitoring of product safety and regulatory compliance throughout the food supply chain is a significant use case for blockchain [Caro et al., 2018]. Many stakeholders, including farmers, bulk milk distributors, manufacturers, wholesalers, and retailers, as shown in Figure 3, are responsible for managing the supply chain operation from the start, when a cow on a farm produces raw milk to the finished product when a consumer buys baby-milk powder.

Internal traceability refers to the traceability of one of the actors' internal processes, whereas chain traceability refers to the traceability of the entire supply chain [Moe, 1998]. To retrieve and provide information to the Food Safety Information System, actors in the dairy supply chain can employ IoT devices and location-based food safety information systems (FSIS) technology. The latter contains a wide range of data that food supply chain actors require to achieve transparency and quality assurance. According to [Aung and Chang, 2014], FSIS is run by unspecified centralized or decentralized information. Each actor is expected to be responsible for food safety in their operations. The Food Safety and Quality Assurance System (FSQAS) establishes the quality and safety standards to which all stakeholders in the supply chain must adhere. The FSIS monitors traceability data to ensure that rules are followed.

This paper focuses on cross-organizational collaboration within the dairy supply chain. Farmers keep detailed records of their farm's location, breed of cow, vaccinations, treatments, and any special regimens that might be required. RFID devices and other sensor networks incorporate blockchain to monitor the health and movement of animals. The health and movement data of animals stored on a blockchain. The bulk milk distribution company is informed through a blockchain-based system when the milk is ready for collection. Temperature control during transit is critical for preventing milk spoilage, and sensors are used to achieve this. GPS technology is often used to monitor vehicles in real-time. When the milk is dispatched to the factory, key information is updated on the blockchain network. This data includes information such as the unit's location (e.g., milk), the number of deliveries at a specific lot, and so forth. The factory processes the milk and manufactures baby milk powder. In addition, consumers are provided with factual data about food items, such as nutritional information, ingredients, expiry date, instructions for use, and other helpful or legally-mandated information.

According to [Casino et al., 2019], SCs are required for food supply-chain operations to improve DAO governance. The supply-chain operation procedures in the food safety and quality assurance system are designed to trigger important events. If, for example, the bulk milk distributor fails to deliver milk to the factory within a specified time or at a specified quality, they may pay the penalty. If this scenario is considered to be important it should be coded into DAO governance. In a traditional supply chain, the parties who work collaboratively often have little control over any organization, which could cause a bottleneck in the system. Still, in a SC-driven blockchain, each business could monitor and track the status of products and transactions with oracles of diverse types, thus providing critical oversight of the whole process. Any bottleneck is immediately visible this way, and the erring party is identified.

Despite these advantages, there are significant business- and legal concerns relating to the fact that blockchain technology is still in the early stages of development, as SCLs are not mature enough to deal with practical realities. Assume, for example, a farmer is managing a workflow process on behalf of the milk processing factory. Even though the

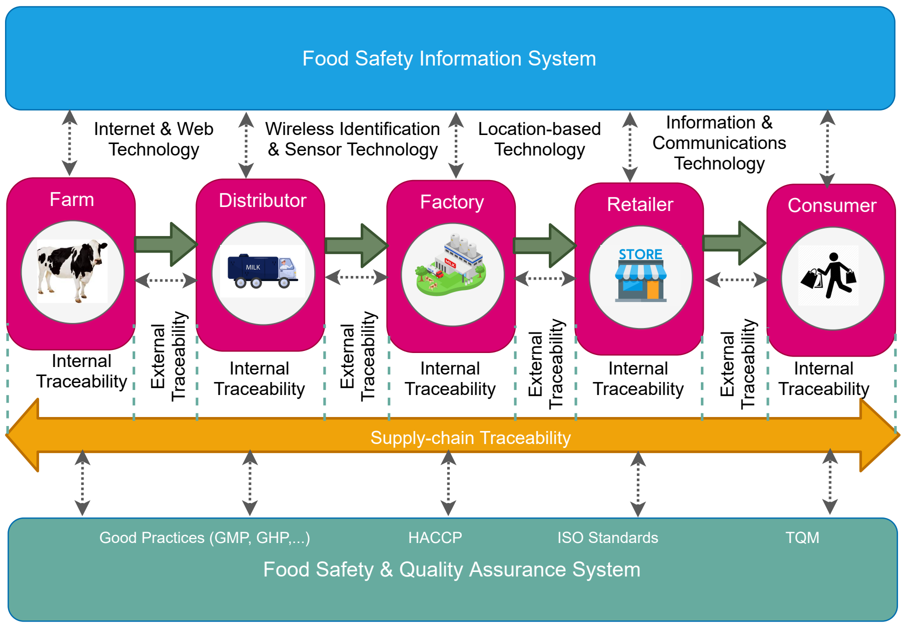
Food Safety Information System
Internet & Web Technology
Wireless Identification & Sensor Technology
Location-based Technology
Information & Communications Technology
Farm
Distributor
Factory
Retailer
Consumer
MILK
STORE
Internal Traceability
External Traceability
Supply-chain Traceability
Good Practices (GMP, GHP,...)
HACCP
ISO Standards
TQM
Food Safety & Quality Assurance System

Figure 3: Use case of the dairy food supply chain, adopted from [Dwivedi and Norta, 2022])

farmer is supplying bulk milk to the factory under a supply contract, he may not wish to reveal the details of all the workflow processes he employs on his farm because they are no concern to the factory. On the one hand, the factory may believe that because they are the farmer's customers, they have a right to have insight into what happens on the farm. The farmer only discloses those aspects of the process he is willing to make public that is of interest to potential customer organizations. On the other hand, most customer organizations aim to incorporate outsourced processes into their processes to provide them with more information about the structure and progress of the process that another company is carrying out.

Still, in reality, the customer organization does not need to know the specifics of the upstream workflow process and only requires a broad understanding. To accommodate such issues, SCs must include clauses that clearly define the rules for when confidential workflow information should be withheld or disclosed to third parties.

4.2 SLCML Instantiation

This section discusses SLCML instantiation corresponding to the running case written by participants. The configuration of rights and obligations for the dairy supply chain running case using the SLCML schema from the previous section. Listing 6 defines the fundamental contractual elements required for any legally-binding business-oriented smart contract. To resolve any conflict, the producer (factory) and distributor create a smart contract with a unique ID that cannot be changed during contract enforcement. Lines 2 and 6 contain the public keys for the producer and the milk distributor, respectively. The parties' names to the contract are specified on Lines 3 and 7. Lines 4 and 8 define the contracting parties' roles, namely: producer as a service consumer and distributor as a milk supplier. On line 10, the contract consideration (milk) for which the parties have agreed to a contractual relationship is listed. The terms and conditions include the obligations and rights outlined in Listing 7 and Listing 8.

```
1  <contract contract_id=”Id1”>
2      <party address=”03 m6”>
3          <name> Producer </name>
4          <role> Service consumer </role>
5      </party>
6      <party address=”31 x7”>
7          <name> Distributor </name>
8          <role> Milk supplier </role>
9      </party>
10     <consideration> Milk </consideration>
11         <terms_and_conditions/>
12     <obligation/>
13     <right/>
14     <prohibitions/>
15     <terms_and_conditions>
16 </contract>
```

Listing 6: Contract instantiation for the dairy supply chain

Listing 7 depicts a producer's commitment to compensate a distributor for milk. The obligation has a name and a unique ID that is used to track performance and is classified as a monetary obligation because of pertaining to economic or financial consequences. Line 3 begins the obligation state, indicating that the producer collects milk in accordance with the orders and is required to pay the distributor money. The producer is the obligor and responsible for carrying out the obligation stated in Line 6. Line 5's obligations benefit the distributor, and we assumed no intermediaries or arbitrators are involved, as indicated by Line 7. The producer is expected to act by paying money, and the to-do obligation (Line 10) has legal consequences. Line 12 implies the obligations for which the producer and distributor sign contracts (Act 1); the producer receives milk from the distributor. The performance type (Line 13) refers to the amount of money that must be transferred from the producer's wallet address to the distributor's wallet address.

In addition, the performance object (Line 14) is defined as a qualified purchase for which a specific amount is compensated within a specific time frame. Line 15 specifies the purchase-payment plan, while the rule conditions specify the payment time limit. Finally, the obligation is amended to include a mention of the existence of a late payment remedy (Line 17). If the producer fails to pay the money within the specified time frame, the producer must transfer interest for the late payment to the distributor.

```
1 <obligation_rule tag_name =”paying_invoices” rule_id =”0001”
2 changeable =”false” monetary =”true”>
3 <state> enabled </state>
4 <parties>
5    <beneficiary> Distributor (31 x7 ) </beneficiary>
6    <obligor> Producer (03 m6 ) </obligor>
7    <third_party> nil </third_party>
```

```
8  </parties>
9  <obligation_type>
10    <legal_obligation> to-do </legal_obligation>
11 </obligation_type>
12 <precondition> act1 (signed)& Milk (transferred) </
     precondition>
13 <performance_type>
14     payment (03 m6,31 x7, buy)
15 </performance_type>
16 <performance_object> invoice ( buy, amount)<
     performance_object>
17 <rule_conditions>
18     date ( before delivery of milk)
19 </rule_conditions>
20 <remedy>late_payment_interest (amount,03 m6 ,31 x7)</remedy>
21 </obligation_rule>
```

Listing 7: Paying milk obligation illustration

The obligation intersects with provisions in the Listing 8 code extract. Because the parties' rights and obligations are intertwined, the other must comply if one party asserts its rights. The rights have a beneficiary who can benefit from them and an obligor who can enable them, as in Listing 7. If the producer receives poor-quality milk, they have the right to replacement. As a result, the distributor will have to replace the milk.

```
1  <right_rule tag_name ="milk_replacement" rule_id ="0002"
2  changeable ="true" monetary ="false">
3  <state> enabled </state>
4  <parties>
5     <beneficiary> Producer (03 m6 ) </beneficiary>
6     <obligor> Distributor (31 x7) </obligor>
7     <third_party> nil </third_party>
8  </parties >
9  <right_type>
10    <conditional_right> claim </conditional_right>
11 </right_type>
12 <precondition> act1 (signed)& Milk (transferred)</
     precondition>
13 <performance_type>
14     replace (poor-quality milk)
15 </performance_type>
16 <action_object>
17     milk (cans of milk, type, and batch unit)
18 </action_object>
19 <rule_conditions> deadline (date) </rule_conditions>
20 <remedy> late_replacement_interest (amount, 31 x7) </remedy>
21 </right_rule>
22
```

Listing 8: Replacing low-quality milk with this example

It is assumed the rights defined in Line 1 have a name and an ID. Because the distributor has the right to revoke the right, he or she can persuade the producer that the quality of the milk was ruined during logistics due to a faulty sensor machine that was not his fault. If the distributor agrees to replace the milk, the contract's rights can be changed while it is being carried out, and the compensation can be set to false. The parties are similarly stated to be in Listing 7, and the right state is ready to be implemented immediately. Because the producer demands that the milk be replaced, the right type is assigned conditional-right. Before the right can be exercised, the contract must be signed and the milk delivered to the producer. The performance type has been changed to cans of milk, type, and batch unit to replace the milk mentioned in the performance object. The corresponding obligation of the distributor must be met within the 'timeframe' specified after this right is activated; otherwise, the producer is entitled to monetary compensation.

4.3 Semantic and Pragmatic Evaluation

Brandtner et al. describe a method for evaluating the syntactic, semantic, and pragmatic aspects of a business-modeling framework for implementing business innovations [Brandtner and Helfert, 2018]. The method is modified and applied to blockchains and for creating SCs domains. We utilize the evaluation approach from Brandtner et al. that provides the evaluation criteria for accessing the semantic and pragmatic usefulness of SLCML. For example, the first column of Table 2 and 3 shows the assessment criteria. The second column shows the question posed to experts for a rating of each assessment criterion. The last column shows the adaptation of the question by introducing the concepts of SLCML into the question. The adapted question in the last column is then posed to the workshop participants and thereby provides the basis for assessing the semantics and pragmatic properties of the SLCML.

The syntactic parts of the SLCML schema have already been specified and evaluated in the previous section 4.2. This section considers semantic and pragmatic aspects, with semantic quality determining how well the new SLCML schema captures contractual business features that domain experts believe are important in describing the domain. Table 2 shows the properties for measuring the semantic aspects of a new modeling language and their application to this paper. Among these properties are the modeling language's validity, relevance to the problem domain, completeness in describing the domain, and language authenticity. The pragmatic aspect evaluates the modeling language's perceived utility in assisting with the design, and implementation, of blockchain-enabled SCs for inter-organizational collaborations. Table 3 shows the properties for measuring the pragmatic aspects of a new modeling language and its adaptation to this paper. These characteristics are the subjective norm, image, job relevance, output quality, result demonstrability, performance, productivity, and perceived usefulness. Experts in the field of blockchain-system design have taken part in evaluating both the semantic and pragmatic aspects of the new modeling language.

We investigate the semantic and pragmatic properties of the SLCML using the modeling language evaluation approach described in Tables 2 and 3, in Section 4.3. The results of the SLCML evaluation are presented in Figure 4. The semantic characteristics are shown on the left side of the figure, and the pragmatic characteristics are depicted as a perceived utility on the right side.

On a scale of 1 to 5, blockchain domain experts agreed that the SLCML schema depicted the design process of legally-binding SCs for semantic quality evaluation appropriately and realistically. They also agree that the SLCML schema accurately and comprehensively represents all of the elements required to create legally enforceable SCs. The average scores for the properties are 4.3 for realisticness, 4.3 for completeness, 4.1 for relevance, and 4.5 for accuracy. The usefulness of the SLCML schema in blockchain-related jobs, the creation of legally-binding SCs output, and improved performance in SCs implementation are all rated highly (above 4.3). The SLCML schema's increased productivity and communication, both with scores of 4.1, are two more pragmatic aspects of the SLCML schema that received high marks. The subjective norm

Title	Description	Adaptation
Correctness	All statements in the representation	SLCML correctly represents the process and elements of creating legally-binding SCs.
Relevance	All statements in the representation are relevant to the problem	SLCML elements are relevant for creating legally-binding SCs.
Completeness	The representation contains all statements about the domain that are correct and relevant	SLCML represents the elements and processes involved in creating legally-binding SCs.
Authenticity	The representation gives a true account of the domain	SLCML represents the elements for creating legally-binding SCs.

Table 2: Assessing SLCML using semantic qualities

property, which measures how important, people in the blockchain community regard the SLCML, is also given a high score of 4.0. According to this assessment, it can be concluded blockchain domain experts agree the SLCML schema is semantically valid and pragmatically beneficial in the construction and development of legally-binding SCs.

Similar scores are recorded by academic and industry professionals in providing details on the semantic quality of SLCML. Academic and industry experts rate the semantic quality of the SLCML at 4.28 and 4.27, respectively. Furthermore, for SLCML pragmatic aspects, domain specialists give similar ratings. The average pragmatic quality of the SLCML is rated 4.06 by academic experts and 4.07 by industry professionals. These results show that academics and industry professionals have similar perceptions of the semantic and pragmatic features of SLCML.

Title	Explanation
Subjective Norm	People who are important will support using SLCML in creating legally-binding SCs.
Image	People in my organization who use SLCML to instantiate SCs would have a high profile.
Job relevance	SLCML usage or application will be relevant in my job.
Output quality	The quality of output I get from SLCML will be high.
Demonstrability	I believe I could explain the benefits of using SLCML to others.
Performance	My job's performance will improve if I use SLCML.
Productivity	I will be more productive if I use the SLCML in my job.
Usefulness	I find the creation of SCs through SLCML is useful in my job.

Table 3: Assessing SLCML using Pragmatic qualities

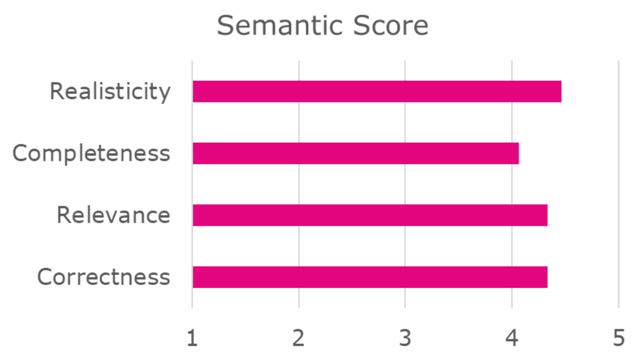

Figure 4: SLCML semantic evaluation result

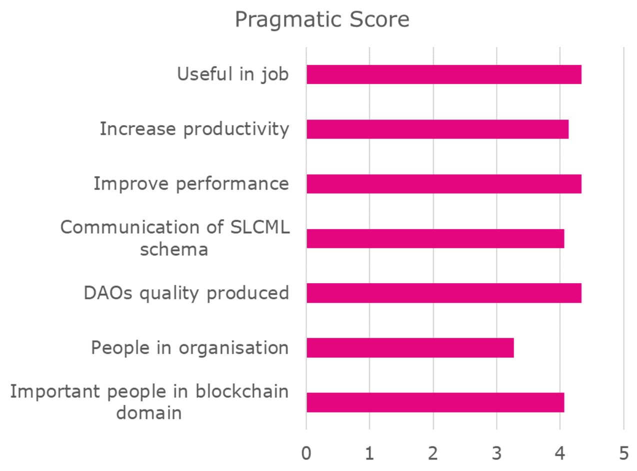

Figure 5: SLCML pragmatic evaluation result

4.4 Usability Evaluation

The paper [Mahunnah et al., 2018], investigates the usability of a support tool for an agent-oriented modeling language. The evaluation method compares the results obtained while using the assistance language tool support with a free-hand sketch. The goal was to assess the validity of the models developed, as well as the time spent developing the models, to determine the benefits of the support tool.

The properties for analyzing the usability of modeling SLCML SCs are shown in Table 4 as an adaptation from [Mahunnah et al., 2018] for this paper. The items assessed the difficulties, time spent, and effort expended in generating SLCML contracts. Other aspects examined include a grasp of the case modeled, comprehension of the SLCML notions, and the application of SLCML in practice. The participants involved in this evaluation are newcomers to the blockchain.

To evaluate the usability of the SLCML, we conduct a workshop with seven participants who are not blockchain experts but can grasp the concepts of SCs design. The participants are master's degree students currently working on topics related to blockchain. During the workshop, the participants receive instruction on the fundamental concepts of the SLCML. The participants are then divided into two groups to model a set of legally-binding SCs using the SLCML. They are tasked with creating an example of an SC for a specific use case in the automobile supply chain. The first group, comprising four students, works with the proposed SLCML, while the second group of three students works with an existing smart-contract modeling language (DAML). A feedback form is used to record the accuracy of the models produced, the amount of time spent, and the ease with which the assignment is completed. The participants are asked to evaluate the SLCML on a scale of 1 to 5.

The usability of developing legally-binding SCs using the SLCML is investigated using the modeling-language support-tool evaluation approach (Table 4). Usability is determined by comparing feedback data from students who develop SCs using SLCML to students who develop SCs using the existing SC modeling language (DAML). Figure 6 depicts the results, where the red bars show the average scores of students who use the existing modeling language, while the blue bars show the average scores of students who use the SLCML schema.

#	Description
P1	The description of the case study was clear to me.
P2	Difficulties in modeling the legal and business requirements in SLCML.
P3	Difficulties in choosing the business processes in SLCML.
P4	Difficulties in modeling the use case instantiation of SLCML.
P5	Short time is required for accomplishing the modeling SCs.
P6	SLCML schema was very useful in modeling legally-binding SCs.
P7	The concepts of the SLCML schema were detailed enough to instantiate the requirements of a blockchain system.
P8	The effort of modeling SCs seems too high for the efficient use of the methodology in practice.

Table 4: Properties for assessing usability of SLCML [Mahunnah et al., 2018]

The results (Fig. 6) show that the students who attend the workshop develop a similar understanding of SLCML concepts and the running scenario. Still, the amount of work required to recreate the modeling SCs and the ease with which the SLCML can be used varies from student to student. Compared to students who use an existing SC modeling language that requires a significant amount of effort to complete the task, students who use the SLCML believe the effort required to create the legally-binding SCs is quite low. Students who use the SLCML enjoy creating legally-binding SCs, while students who use the existing language find it difficult to model legally-binding SCs. The findings show that the proposed SLCML can be used to easily design and develop legally enforceable documents.

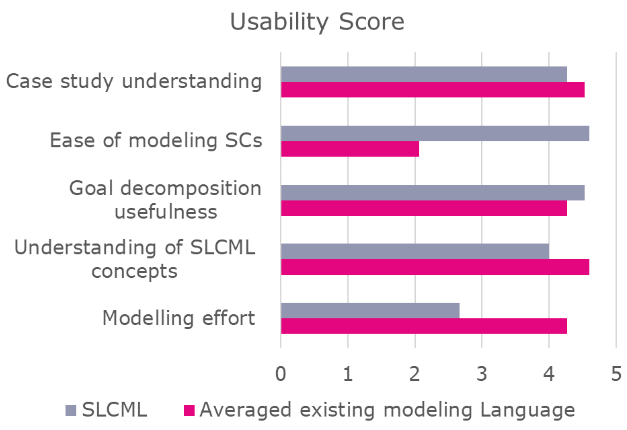

Figure 6: SLCML usability evaluation result

5 Discussion and Future Work

The evaluations conducted to assess the main artifacts produced in this paper are detailed in Section 4. The SLCML schema is evaluated for its suitability for generating legally-binding SCs. The SLCML schema is evaluated to determine its practicality and usefulness using the method described in [Brandtner and Helfert, 2018]. The SLCML's effectiveness is determined by its usability and ease of generating error-free SCs. This is accomplished by employing the method outlined in [Mahunnah et al., 2018].

According to the evaluation results, the modeling language is highly realistic for describing the legal and business elements necessary for creating legally-binding SCs. According to the practical usefulness results, the SLCML schema is suitable for producing high-quality legally-binding SCs. Thus, significantly improving the performance and productivity of analysts and developers in building SCs. Furthermore, the SLCML demonstrates applicability and utility in tasks involving the design and development of blockchain-based SCs. Some similarities and differences emerge when these findings are compared to those of a similar study [He et al., 2018], which demonstrates a process-based approach for modeling innovations in organizations. The results of [He et al., 2018] show that the modeling approach is relevant for innovation management tasks and has high practical usefulness. Still, compared to the other results, the average semantic qualities score is low, falling just above the second quartile (above 50%).

In comparison, the average score for the modeling approach's semantic and practical usefulness for the proposed SLCML schema is in the third quartile (above 75%). This demonstrates that the SLCML is not only useful for creating blockchain-based SCs, but it also accurately represents legal- and business-related elements and processes in the blockchain domain. In addition, the results of the SLCML usability evaluation show that the SLCML is highly practical and usable in producing correct SLCML contracts compared to a freehand smart-contract modeling language. The results show that the modeling effort required to create SCs with SLCML is relatively low, whereas the effort required to create SCs with the existing modeling language is exceptionally high. The SLCML is simple to model and ranks in the third quartile (above 50%), while the existing modeling language (SPESC) ranks in the second quartile (below 50%). This is similar to the results of this work, demonstrating that SLCML is more user-friendly than an existing modeling language.

This paper has several *limitations*, which we discuss here. For example, the researcher's subjectivity in analyzing and interpreting data and the risk of generalization. Subjectivity related to the researcher's values and viewpoints can impact the interpretation of results when analyzing the strengths and weaknesses of current approaches for building the SCL. Subjectivity may also impact how the SLCML evaluation results are interpreted. Nonetheless, blockchain technology is in its infancy, and the number of experts with the necessary knowledge was limited to participate in the webinar and workshop held as part of this research. As a result, assembling many experts to participate in several of the paper's evaluations is a significant constraint. We performed the evaluation in a controlled environment, which does not adequately reflect the conditions under which the proposed evaluation approaches will be applied in the real world. As previously stated, we organized a one-time webinar event and workshop, and it was difficult to follow up with participants to collect additional data for further analysis. Addressing the limitations discussed above can lead to a more comprehensive understanding of the results of this work.

6 Conclusion

The primary artifacts created during the research study are evaluated in this paper. One of the artifacts is the SLCML schema, which contains legal and business semantics for designing and developing legally-binding SCs. To this end, we examine the literature for suitable methods to evaluate the SLCML. The evaluation methods are compared to see their syntactic accuracy, semantic correctness, and utility of a modeling language. Based on the analysis results, appropriate SLCML evaluation techniques tailored to the current paper are chosen. The SLCML evaluation seeks to determine the semantic quality and practical utility of the SLCML in creating legally-binding SCs. The evaluation also determines the SLCML's effectiveness in producing SCs. The semantic and pragmatic evaluation results, as well as the usability evaluation results, are presented separately.

The main artifacts produced in this paper are 1) The proposed SLCML; an XML-based language, 2) the demonstration of smart contract development based on the proposed SLCML language, 3) the evaluation of the SLCML schema's utility in the creation of legally-binding SCs.

The lack of domain completeness is the main limitation of this research. The SLCML's development assistance in generating legally-binding SCs is limited to business-to-business (B2B) contracts. Other study limitations include the researcher's subjectivity in analyzing and interpreting data and the risk of generalization. Subjectivity related to the researcher's values and viewpoints could impact the interpretation of results when analyzing the strengths and weaknesses of current approaches for building the SCL. Subjectivity may also have an impact on how the SLCML evaluation results are interpreted. The results of this paper may be generalizable due to the large number of experts interviewed in this study.

Acknowledgements

This project is partially funded by the Estonian personal research funding "Team grant (PRG) project PRG1641".

References

[Abodei et al., 2019a] Abodei, E., Norta, A., Azogu, I., Udokwu, C., and Draheim, D. (2019a). Blockchain technology for enabling transparent and traceable government collaboration in public project processes of developing economies. In *Lecture Notes in Computer Science*, pages 464–475. Springer International Publishing.

[Abodei et al., 2019b] Abodei, E., Norta, A., Azogu, I., Udokwu, C., and Draheim, D. (2019b). Blockchain technology for enabling transparent and traceable government collaboration in public project processes of developing economies. In *Lecture Notes in Computer Science*, pages 464–475. Springer International Publishing.

[Aung and Chang, 2014] Aung, M. M. and Chang, Y. S. (2014). Traceability in a food supply chain: Safety and quality perspectives. *Food Control*, 39:172–184.

[Bartoletti and Pompianu, 2017] Bartoletti, M. and Pompianu, L. (2017). An empirical analysis of smart contracts: platforms, applications, and design patterns. In *International conference on financial cryptography and data security*, pages 494–509. Springer.

[Behnke and Janssen, 2020] Behnke, K. and Janssen, M. (2020). Boundary conditions for traceability in food supply chains using blockchain technology. *International Journal of Information Management*, 52:101969.

[Brandtner and Helfert, 2018] Brandtner, P. and Helfert, M. (2018). Multi-media and web-based evaluation of design artifacts-syntactic, semantic and pragmatic quality of process models. *Systems, Signs and Actions: An International Journal on Information Technology, Action, Communication and Workpractices*, 11(1):54–78.

[Caro et al., 2018] Caro, M. P., Ali, M. S., Vecchio, M., and Giaffreda, R. (2018). Blockchain-based traceability in agri-food supply chain management: A practical implementation. In *2018 IoT Vertical and Topical Summit on Agriculture - Tuscany (IOT Tuscany)*, pages 1–4.

[Casino et al., 2019] Casino, F., Kanakaris, V., Dasaklis, T. K., Moschuris, S., and Rachaniotis, N. P. (2019). Modeling food supply chain traceability based on blockchain technology. *IFAC-PapersOnLine*, 52(13):2728–2733. 9th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2019.

[Crawford and Ostrom, 1995] Crawford, S. E. S. and Ostrom, E. (1995). A grammar of institutions. *American Political Science Review*, 89(3):582–600.

[Croman et al., 2016] Croman, K., Decker, C., Eyal, I., Gencer, A. E., Juels, A., Kosba, A., Miller, A., Saxena, P., Shi, E., Gün Sirer, E., Song, D., and Wattenhofer, R. (2016). On scaling decentralized blockchains. In Clark, J., Meiklejohn, S., Ryan, P. Y., Wallach, D., Brenner, M., and Rohloff, K., editors, *Financial Cryptography and Data Security*, pages 106–125, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Dannen, 2017] Dannen, C. (2017). *Introducing Ethereum and solidity*, volume 318. Springer.

[de Kruijff and Weigand, 2017] de Kruijff, J. and Weigand, H. (2017). Ontologies for commitment-based smart contracts. In Panetto, H., Debruyne, C., Gaaloul, W., Papazoglou, M., Paschke, A., Ardagna, C. A., and Meersman, R., editors, *On the Move to Meaningful Internet Systems. OTM 2017 Conferences*, pages 383–398, Cham. Springer International Publishing.

[Dixit et al., 2022] Dixit, A., Deval, V., Dwivedi, V., Norta, A., and Draheim, D. (2022). Towards user-centered and legally relevant smart-contract development: A systematic literature review. *Journal of Industrial Information Integration*, 26:100314.

[Dranidis, 2007] Dranidis, D. (2007). Evaluation of studentuml: an educational tool for consistent modelling with uml. In *Proceedings of the Informatics Education Europe II Conference*.

[Durov, 2019] Durov, N. (2019). Fift : A brief introduction. Technical report.

[Dwivedi and Norta, 2021] Dwivedi, V. and Norta, A. (2021). A legal-relationship establishment in smart contracts: Ontological semantics for programming-language development. In Singh, M., Tyagi, V., Gupta, P. K., Flusser, J., Ören, T., and Sonawane, V. R., editors, *Advances in Computing and Data Sciences*, pages 660–676, Cham. Springer International Publishing.

[Dwivedi and Norta, 2022] Dwivedi, V. and Norta, A. (2022). Auto-generation of smart contracts from a domain-specific xml-based language. In Satapathy, S. C., Peer, P., Tang, J., Bhateja, V., and Ghosh, A., editors, *Intelligent Data Engineering and Analytics*, pages 549–564, Singapore. Springer Singapore.

[Dwivedi et al., 2021a] Dwivedi, V., Norta, A., Wulf, A., Leiding, B., Saxena, S., and Udokwu, C. (2021a). A formal specification smart-contract language for legally binding decentralized autonomous organizations. *IEEE Access*, 9:76069–76082.

[Dwivedi et al., 2021b] Dwivedi, V., Pattanaik, V., Deval, V., Dixit, A., Norta, A., and Draheim, D. (2021b). Legally enforceable smart-contract languages: A systematic literature review. *ACM Comput. Surv.*, 54(5).

[Dwivedi, 2022] Dwivedi, V. K. (2022). *A Legally Relevant Socio-Technical Language Development for Smart Contracts*. PhD thesis.

[Farrell et al., 2017] Farrell, S., Machin, H., and Hinchliffe, R. (2017). Lost and found in smart contract translation—considerations in transitioning to automation in legal architecture. In *UNCITRAL, Modernizing international trade law to support innovation and sustainable development. Proceedings of the congress of the United Nations commission on international trade law*, volume 4, pages 95–104.

[Filippi and Hassan, 2016] Filippi, P. D. and Hassan, S. (2016). Blockchain technology as a regulatory technology: From code is law to law is code. *First Monday*.

[Frantz and Nowostawski, 2016] Frantz, C. K. and Nowostawski, M. (2016). From institutions to code: Towards automated generation of smart contracts. In *2016 IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS*W)*, pages 210–215.

[George and Lesaege, 2020] George, W. and Lesaege, C. (2020). A Smart Contract Oracle for Approximating Real-World, Real Number Values. In Danos, V., Herlihy, M., Potop-Butucaru, M., Prat, J., and Tucci-Piergiovanni, S., editors, *International Conference on Blockchain Economics, Security and Protocols (Tokenomics 2019)*, volume 71 of *OpenAccess Series in Informatics (OASIcs)*, pages 6:1–6:15, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[Goldenfein and Leiter, 2018] Goldenfein, J. and Leiter, A. (2018). Legal engineering on the blockchain: 'smart contracts' as legal conduct. *Law and Critique*, 29(2):141–149.

[Governatori et al., 2018] Governatori, G., Idelberger, F., Milosevic, Z., Riveret, R., Sartor, G., and Xu, X. (2018). On legal contracts, imperative and declarative smart contracts, and blockchain systems. *Artificial Intelligence and Law*, 26(4):377–409.

[Halvorsrud et al., 2016] Halvorsrud, R., Haugstveit, I. M., and Pultier, A. (2016). Evaluation of a modelling language for customer journeys. In *2016 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 40–48. IEEE.

[He et al., 2018] He, X., Qin, B., Zhu, Y., Chen, X., and Liu, Y. (2018). Spesc: A specification language for smart contracts. In *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, volume 01, pages 132–137.

[He et al., 2018] He, X., Qin, B., Zhu, Y., Chen, X., and Liu, Y. (2018). Spesc: A specification language for smart contracts. In *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, volume 01, pages 132–137.

[Home, 2005] Home, A. (2005). Generate anything from any emf model.

[Howson, 2020] Howson, P. (2020). Building trust and equity in marine conservation and fisheries supply chain management with blockchain. *Marine Policy*, 115:103873.

[Iqbal and Matulevičius, 2021] Iqbal, M. and Matulevičius, R. (2021). Exploring sybil and double-spending risks in blockchain systems. *IEEE Access*, 9:76153–76177.

[Jaccheri and Stålhane, 2001] Jaccheri, M. L. and Stålhane, T. (2001). Evaluation of the e3 process modelling language and tool for the purpose of model creation. In *International Conference on Product Focused Software Process Improvement*, pages 271–281. Springer.

[Jain et al., 2018] Jain, A., Arora, S., Shukla, Y., Patil, T., and Sawant-Patil, S. (2018). Proof of stake with casper the friendly finality gadget protocol for fair validation consensus in ethereum. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 3(3):291–298.

[Khalil et al., 2017] Khalil, F. A., Butler, T., O'Brien, L., and Ceci, M. (2017). Trust in smart contracts is a process, as well. In *Financial Cryptography and Data Security*, pages 510–519. Springer International Publishing.

[Kormiltsyn et al., 2019] Kormiltsyn, A., Udokwu, C., Karu, K., Thangalimodzi, K., and Norta, A. (2019). Improving healthcare processes with smart contracts. In Abramowicz, W. and Corchuelo, R., editors, *Business Information Systems*, pages 500–513, Cham. Springer International Publishing.

[Lin et al., 2018] Lin, J., Shen, Z., Zhang, A., and Chai, Y. (2018). Blockchain and iot based food traceability for smart agriculture. In *Proceedings of the 3rd International Conference on Crowd Science and Engineering*, ICCSE'18, New York, NY, USA. Association for Computing Machinery.

[López-Pintado et al., 2019] López-Pintado, O., García-Bañuelos, L., Dumas, M., Weber, I., and Ponomarev, A. (2019). Caterpillar: A business process execution engine on the ethereum blockchain. *Software: Practice and Experience*, 49(7):1162–1193.

[Mahunnah et al., 2018] Mahunnah, M., Taveter, K., and Matulevičius, R. (2018). An empirical evaluation of the requirements engineering tool for socio-technical systems. In *2018 IEEE 7th International Workshop on Empirical Requirements Engineering (EmpiRE)*, pages 8–15. IEEE.

[Matulevičius et al., 2017] Matulevičius, R., Norta, A., Udokwu, C., and Nõukas, R. (2017). Assessment of aviation security risk management for airline turnaround processes. In Hameurlain, A., Küng, J., Wagner, R., Dang, T. K., and Thoai, N., editors, *Transactions on Large-Scale Data- and Knowledge-Centered Systems XXXVI: Special Issue on Data and Security Engineering*, pages 109–141, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Mavridou and Laszka, 2017] Mavridou, A. and Laszka, A. (2017). Designing secure ethereum smart contracts: A finite state machine based approach.

[Mavridou and Laszka, 2018] Mavridou, A. and Laszka, A. (2018). Tool demonstration: Fsolidm for designing secure ethereum smart contracts. In Bauer, L. and Küsters, R., editors, *Principles of Security and Trust*, pages 270–277, Cham. Springer International Publishing.

[Mavridou et al., 2019] Mavridou, A., Laszka, A., Stachtiari, E., and Dubey, A. (2019). Verisolid: Correct-by-design smart contracts for ethereum.

[Mendling et al., 2018] Mendling, J., Weber, I., Aalst, W. V. D., Brocke, J. V., Cabanillas, C., Daniel, F., Debois, S., Ciccio, C. D., Dumas, M., Dustdar, S., Gal, A., García-Bañuelos, L., Governatori, G., Hull, R., Rosa, M. L., Leopold, H., Leymann, F., Recker, J., Reichert, M., Reijers, H. A., Rinderle-Ma, S., Solti, A., Rosemann, M., Schulte, S., Singh, M. P., Slaats, T., Staples, M., Weber, B., Weidlich, M., Weske, M., Xu, X., and Zhu, L. (2018). Blockchains for business process management - challenges and opportunities. *ACM Trans. Manage. Inf. Syst.*, 9(1).

[Moe, 1998] Moe, T. (1998). Perspectives on traceability in food manufacture. *Trends in Food Science & Technology*, 9(5):211–214.

[Morandini et al., 2011] Morandini, M., Perini, A., and Marchetto, A. (2011). Empirical evaluation of tropos4as modelling. *iStar*, 766:14–19.

[Mülder, 2019] Mülder, A. (2019). Model-driven smart contract development for everyone.

[Nakamoto, 2008] Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. Technical report.

[Norta, 2015] Norta, A. (2015). Creation of smart-contracting collaborations for decentralized autonomous organizations. In Matulevičius, R. and Dumas, M., editors, *Perspectives in Business Informatics Research*, pages 3–17, Cham. Springer International Publishing.

[Opdahl and Henderson-Sellers, 2002] Opdahl, A. L. and Henderson-Sellers, B. (2002). Ontological evaluation of the uml using the bunge–wand–weber model. *Software and systems modeling*, 1(1):43–67.

[Orlenyslp, 2019] Orlenyslp (2019). orlenyslp/caterpillar.

[Pilkington, 2016] Pilkington, M. (2016). Blockchain technology: principles and applications.

[Pourmirza et al., 2019] Pourmirza, S., Peters, S., Dijkman, R., and Grefen, P. (2019). Bpms-ra: A novel reference architecture for business process management systems. *ACM Trans. Internet Technol.*, 19(1).

[Savelyev, 2017] Savelyev, A. (2017). Contract law 2.0: 'smart' contracts as the beginning of the end of classic contract law. *Information & Communications Technology Law*, 26(2):116–134.

[Sergey et al., 2019] Sergey, I., Nagaraj, V., Johannsen, J., Kumar, A., Trunov, A., and Hao, K. C. G. (2019). Safer smart contract programming with scilla. *Proc. ACM Program. Lang.*, 3(OOPSLA).

[Shekhar and Xiong, 2008] Shekhar, S. and Xiong, H. (2008). Agent-based models. In *Encyclopedia of GIS*, pages 11–11. Springer US.

[Smajgl et al., 2010] Smajgl, A., Izquierdo, L. R., and Huigen, M. (2010). Rules, knowledge and complexity: How agents shape their institutional environment. *Journal of Modelling & Simulation of Systems*, 1(2).

[Swan, 2015] Swan, M. (2015). *Blockchain: Blueprint for a new economy.* " O'Reilly Media, Inc.".

[Syahputra and Weigand, 2019] Syahputra, H. and Weigand, H. (2019). The development of smart contracts for heterogeneous blockchains. In Popplewell, K., Thoben, K.-D., Knothe, T., and Poler, R., editors, *Enterprise Interoperability VIII*, pages 229–238, Cham. Springer International Publishing.

[Szabo, 1997] Szabo, N. (1997). Formalizing and securing relationships on public networks. *First Monday*, 2(9).

[Tran et al., 2018] Tran, A. B., Lu, Q., and Weber, I. (2018). Lorikeet: A model-driven engineering tool for blockchain-based business process execution and asset management. In *BPM*.

[Valliappan et al., 2018] Valliappan, N., Mirliaz, S., Vesga, E. L., and Russo, A. (2018). Towards adding variety to simplicity. In *Lecture Notes in Computer Science*, pages 414–431. Springer International Publishing.

[Von Alan et al., 2004] Von Alan, R. H., March, S. T., Park, J., and Ram, S. (2004). Design science in information systems research. *MIS quarterly*, 28(1):75–105.

[Weber et al., 2016] Weber, I., Xu, X., Riveret, R., Governatori, G., Ponomarev, A., and Mendling, J. (2016). Untrusted business process monitoring and execution using blockchain. In *Lecture Notes in Computer Science*, pages 329–347. Springer International Publishing.

[Yussupov et al., 2022] Yussupov, V., Soldani, J., Breitenbücher, U., and Leymann, F. (2022). Standards-based modeling and deployment of serverless function orchestrations using bpmn and tosca. *Software: Practice and Experience*.

[Zhou et al., 2017] Zhou, J., Shafique, M. N., Adeel, A., Nawaz, S., and Kumar, P. (2017). What is Theoretical Contribution? A Narrative Review. *Sarhad Journal of Management Sciences*, 3(2):261–271.