

# Blockchain Application Development Using Model-Driven Engineering and Low-Code Platforms: A Survey

Simon Curty<sup>1</sup>[0000–0002–2868–9001], Felix Härer<sup>1</sup>[0000–0002–2768–2342], and  
Hans-Georg Fill<sup>1</sup>[0000–0001–5076–5341]

University of Fribourg, Digitalization and Information Systems Group, Fribourg, CH  
[firstname.lastname@unifr.ch](mailto:firstname.lastname@unifr.ch)  
<https://www.unifr.ch/inf/digits/>

**Abstract.** The creation of blockchain-based software applications requires today considerable technical knowledge, particularly in software design and programming. This is regarded as a major barrier in adopting this technology in business and making it accessible to a wider audience. As a solution, no-code and low-code approaches have been proposed that require only little or no programming knowledge for creating full-fledged software applications. In this paper we review academic approaches from the discipline of model-driven engineering as well as industrial no-code and low-code development platforms for blockchains. We further present a case study for an integrated no-code blockchain environment for demonstrating the state-of-the-art in this area. Based on the gained insights we derive requirements for the future development of no-code and low-code approaches that are dedicated to the field of blockchains.

**Keywords:** Blockchain · Low-Code · No-Code · Model-Driven Engineering · Software Development

## 1 Introduction

With the further maturing of blockchain technologies and the soon expected transition to more energy-efficient and faster protocols with higher transaction volumes [13,11,24], a more widespread adoption of these technologies seems within reach. However, one considerable barrier limiting the adoption is the technical and organizational complexity that users are confronted with when creating blockchain-based applications [18]. This complexity originates on the one hand from the underlying technical foundations, which build on distributed and decentralized systems, cryptography, and algorithmic processing [2]. Blockchains such as Ethereum combine these properties for storing transactions in an append-only data structure, where each new block has a cryptographically verifiable link to its predecessor. Thus, users are part of a decentralized network that minimizes the degree of trust required towards other participants who continuously validate the links of the blockchain. In addition, organizational barriers such as the

involvement of new regulatory requirements, the development of new skills and competencies, and the availability of financial and human resources may prevent adoption in practice [8].

From the perspective of software engineering, the lack of specialists for programming may today be partly compensated with so-called *low-code* platforms [12,31,4]. These development platforms are typically available as cloud services with visual, diagrammatic interfaces and declarative languages. In our view, they constitute the next step in the industry adoption of academic model-driven engineering (MDE) approaches and its predecessors where models are regarded as primary development artifacts for software engineering [36,5,10]. While *low-code* approaches allow a user to produce results without having to understand source code and there may be an underlying model integrated with features of the platform [4], the model may not conform to an explicit formalization [10]. Further, we consider so-called *no-code* approaches as a subset of low-code approaches that operate at an abstraction level above code, not showing code to the user at all. Today, a large number of such platforms and tools are available that either support the development of complete software applications or focus on providing specific functionality, e.g. for entering data in a form and saving it to a database [26].

For easing the creation of blockchain-based applications it seems obvious to revert to MDE and low-code approaches. These carry the potential to abstract from the technical complexity and enable users to focus on usage scenarios and the organizational embedding. In the following we investigate academic and industrial approaches for realizing blockchain applications using these methods. We will do this along the following three research questions. *RQ1*: Which academic MDE approaches exist for the development of blockchain-based applications?, *RQ2*: Which low-code and no-code platforms permit the realization of blockchain-based applications?, *RQ3*: What are requirements for future blockchain development platforms that are informed by MDE, no-code and low-code?

In particular, we will regard approaches that are already available for creating blockchain-based software applications or offer interfaces to other platforms enabling this. This will permit to describe the state-of-the-art in this area and derive requirements for the development of future approaches. The remainder of the paper is structured as follows. Section 2 will outline related work in the form of previous studies and lead over to our research methodology in Section 3. Subsequently, we will present in Section 4 our review of academic MDE approaches and in Section 5 the review of no-code and low-code development platforms used in industry. Section 6 presents a blockchain use case using state-of-the-art low-code platforms, resulting in the discussion of requirements in Section 7.

## 2 Related Studies

Developing blockchain-based applications requires a high level of expertise and understanding of the underlying technologies. Blockchain-based applications are empowered by smart contracts, i.e. programs executed on the blockchain. These

smart contracts often involve financial transactions or deal with issues related to trust. As such, their correctness is of utmost importance. Due to the immutable nature of blockchains, mistakes in smart contract implementations are difficult to rectify. This can be eased through different visual languages for smart contracts, which have been reviewed and compared in [15]. While visual programming languages aim to reduce complexity and improve accessibility for the programmer, they do not correspond in general to low-code development approaches, which may involve visual programming but also deal with the generation and life-cycle management of software artifacts. Approaches and tools for the analysis and development of smart contracts have been reviewed in [19,33]. While both studies discuss issues related to software engineering, such as code analysis and testing, model-driven or low-code techniques to develop blockchain-based software are not regarded.

The study by Ait Hsain et al. [1] focuses on MDE for Ethereum smart contracts, however the review process is not elaborated. Sánchez-Gómez et al. [29] review model-based testing and development approaches. Since the publication of their study, newer approaches have emerged. A more recent review of MDE methods was conducted by Levasseur et al. [21]. In comparison to their work, we applied a broader search methodology and identified more approaches. None of these studies consider industrial approaches such as platforms and focus predominantly on smart contracts.

In summary, while numerous studies on issues regarding smart contract development have been conducted, to the best of our knowledge, a comprehensive review of the state-of-the-art of MDE and low-code/no-code approaches from both academia and industry in this field is missing so far.

### 3 Research Methodology

For answering the three research questions we will employ the following research methodology. At first we review existing academic MDE approaches for blockchain applications in the form of a structured literature review (SLR). Thereby we follow the guidelines by Webster and Watson [35] and vom Brocke et al. [6]. The initial corpus of the SLR was generated by searching all keyword combinations from two groups, where group one included '*blockchain, distributed ledger, smart contract*' and group two '*enterprise model, conceptual model, business model, model-driven, no-code, low-code*'. These keywords were selected based on the domain understanding of the authors. We expected the relevant concepts to be dispersed, thus we chose a broad set of keywords.

For discovering relevant industrial approaches, we reverted to expert knowledge from industry in the field of low-code development combined with our own searches. On this bases, we conducted (1) a survey of available platforms towards suitability for blockchain application development and (2) the implementation of a blockchain use case as an evaluation. This exploratory research approach is directed towards discovering requirements for future platforms that combine

blockchain application development with the state-of-the-art from academia and industry.

## 4 Academic MDE Approaches

In the following subsections, we review approaches of the academic discipline *model-driven engineering* in regard to development solutions for blockchains.

Model-driven engineering introduces models as primary artifact to the software development process in order to address numerous challenges of software engineering [5,27]: First, the common understanding of software artifacts can be facilitated by domain-specific models, as such models are easier to interpret for humans than code. Second, model-based reasoning allows the verification of software, e.g., to determine the fulfillment of security properties. And third, well-defined models allow developers to create software artifacts in an automated fashion, which are correct-by-construction, with no or reduced coding effort. To identify existing MDE approaches that target specifically the development of blockchain applications, we conducted a systematic literature review as elaborated in the following.

### 4.1 Review Process

The systematic review process as shown in Figure 1 follows the guidelines by [35] and [6]. To obtain an initial corpus of publications, we performed keyword searches in step (S-1) on ACM, Springer, and IEEE Explore with the search strings shown in Table 1. From the resulting corpus, duplicates were removed in step (S-2). Due to the large number of documents, we filtered the publications by outlets in step (S-3) that typically publish papers in software engineering, model-driven engineering or information systems. Before the fulltext analysis, the reduced corpus was then screened by titles in step (S-4).

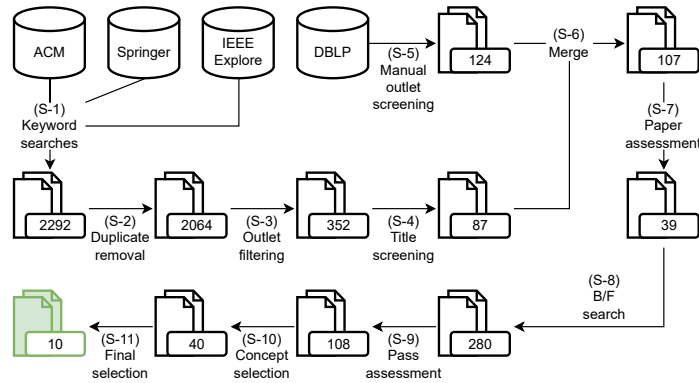


Fig. 1. Academic Literature Review Process

As basis for this fourth step we formulated keyword criteria, whereby the title should contain one of "conceptual", "model", "process", "execution", "process", "architecture", "framework", "design", "development", "pattern", "use case", "supply chain", "database", "storage", "verification", "generation", "language", and mention a blockchain-related word, such as "distributed", "chain", "contract". Additionally, we analyzed all titles to capture promising publications. In parallel, we screened in step (S-5) the table of contents of selected outlets in software engineering and related disciplines by applying the same process as in (S-4). That is, we screened all proceedings, workshops, issues, etc., published in one of the following outlets from 2015 to Nov. 2021: *BCCA*, *BMSD*, *BPMDs/EMMSAD*, *BRAINS*, *COINS*, *CSIMQ*, *CVCBT*, *DAPPCON*, *DK*, *EMISA*, *ER*, *ICBC*, *ICBCT*, *IEEE Blockchain*, *IEEE ICBC*, *IJISMD*, *MoDELS*, *PoEM* and *SoSyM*. These two sets of publications were then merged and duplicates removed (S-6).

Search string	Results
("blockchain" OR "distributed ledger" OR "smart contract" OR "smart-contract") AND ("All "business model" OR "business modeling") AND (year>2014)	1625
("blockchain" OR "distributed ledger" OR "smart contract" OR "smart-contract") AND ("All "enterprise model" OR "enterprise modeling") AND (year>2014)	40
("blockchain" OR "distributed ledger" OR "smart contract" OR "smart-contract") AND ("All "conceptual model" OR "conceptual modeling") AND (year>2014)	370
("blockchain" OR "distributed ledger" OR "smart contract" OR "smart-contract") AND ("All "model driven" OR "model-driven") AND (year>2014)	181
("blockchain" OR "distributed ledger" OR "smart contract" OR "smart-contract") AND ("All "no code" OR "no-code" OR "low code" OR "low-code") AND (year>2014)	76

**Table 1.** Simplified search strings used and results found on ACM, IEEE Explore, and Springer. The concrete syntax of search strings varies for each search portal.

In the next step, the publications were assessed by at least reading the abstract and reviewing tables and images (S-7), considering the inclusion criteria that (i) the publication should be directly related to distributed ledger technologies, and (ii) creates, discusses, or presents a modeling approach. Publications using models to only illustrate software, systems, or a use case, e.g., by means of a standard UML use case diagram, were excluded. For the remaining publications, we then performed a recursive backward-forward search, as proposed in [34] (S-8): references and citations were screened, seemingly relevant publications added to the set, and subsequently assessed as in step (S-7). For all relevant new additions, a backward-forward search was again performed.

Eventually, no new relevant publications could be found and the backward-forward search was concluded. Of all thus collected publications, 108 fulfilled the assessment criteria (S-9). We further filtered by contained concepts in step (S-10), i.e., (i) the approach has MDE characteristics, (ii) it must be tool-assisted, and (iii) include generation of code, application artifacts, or some executable specifications. The motivation for choosing these criteria is founded in the commonalities of low-code/no-code and MDE, as elaborated in Section 1. Finally, we selected 10 approaches we consider representative for the full spectrum of academic approaches.

## 4.2 Results

In Table 2, the final selection of academic approaches from (S-11) is shown. We further evaluated the approaches regarding the required user expertise - see column *Expertise*. Approaches where a user must not write any code and only basic understanding of blockchain concepts is required, we consider suitable for *non-technical* users. In contrast, approaches that require understanding of advanced concepts or chain-specific features, e.g. gas costs in Ethereum, we consider suitable for *non-programmers*. Finally, if the user has to write any code, the approach is only suitable for *programmers*.

For the comparison of the academic approaches, we classified them in addition using three layers, which are based on the traditional layers of ArchiMate<sup>1</sup>. This choice is motivated by a previous application of ArchiMate in the context of Blockchain use cases [9]. Approaches on the **(i) business layer** integrate modeling of business concepts, such as use cases from a top-down perspective. The **(ii) application layer** includes approaches which integrate life-cycle and deployment management, or integration facilities. Finally, on the **(iii) technology layer**, we consider approaches whose scope is limited to the generation of smart contract code from models.

Ref.	Name	BP	Modeling language	Layer	Impl. platform	Expertise	OS
[3]	Archi2HC	H	ArchiMate	Business	Archi	●●●	-
[22]	Caterpillar	E	BPMN	Application	custom (Node.js, bpmn-js)	●●○	+
[17]	ChainOps	E	domain-specific	Application	AstraKode Blockchain Modeler	●●○	-
[28]	Das Contract	E,C	DEMO, BPMN, Blockly	Technology	custom (.Net, Node.js)	●●○	○
[25]	iContractBot	MC	domain-specific (iContractML)	Technology	Xatkit	●●○	○
[14]	iContractML	MC	domain-specific (iContractML)	Technology	Obeo Designer (Eclipse Sirius)	●●○	○
[30]	LATTE	E	domain-specific	Technology	custom (Electron)	●●○	+
[32]	LEMMA	E	domain-specific (LEMMA)	Application	LEMMA	●●●	○
[20]	UML2Go	H	UML	Technology	Obeo Acceleo (Eclipse)	●●○	-
[23]	VeriSolid	E	domain-specific (state machine)	Technology	WebGME	●●○	+

Name: Short name of the approach. If none was given by the authors we assigned one.

BP: Blockchain platform, E: Ethereum, C: Cardano, H: Hyperledger Fabric, MC: Multi-chain

Exp.: Required expertise, ●: non-technical, ●●: non-programmer, ●●●: programmer

OS: open source, +: available, ○: no license specified, -: not available

**Table 2.** Selected academic, model-driven approaches for blockchain application development that apply code generation.

In the entire corpus of publications, we could identify only one approach which clearly lies on the **business layer (i)** and simultaneously permits the generation of code artifacts. Babkin et al. [3] propose a mapping between ArchiMate concepts and Hyperledger Composer constructs. From an ArchiMate model, a

<sup>1</sup> See <https://www.opengroup.org/archimate-forum/archimate-overview>

project artifact for Hyperledger Fabric is generated. However, a programmer must implement the business logic manually. The Das Contract approach [28] applies modeling languages of DEMO to design and generate smart contracts. While DEMO is traditionally used to model organizations, this is however not part of this approach.

On the **application layer (ii)** lie approaches and tools offering integration and management capabilities for the generated artifacts. Caterpillar [22] is a process execution system, in which processes are modeled as BPMN in a web-based visual editor. The models may be translated to Solidity code or into an intermediate representation to be executed by an on-chain execution engine. Furthermore, the tool offers a model repository and monitoring of processes. In the ChainOps [17] framework, smart contracts are composed visually from pre-defined templates, subsequently validated against domain-specific constraints and policies. Models then are sent to a REST service to be translated and deployed. The vision of ChainOps is to offer a complete and integrated Dapp life-cycle solution for the OntoChain<sup>2</sup> ecosystem. The work of Trebbau et al. [32] is an extension of LEMMA, a modeling framework for microservices. Using the modeling languages of LEMMA, code artifacts for the connection to chain networks and smart contract interaction may be generated. The focus of this approach is the model-based integration of on-chain components.

Most identified approaches focus on the generation of smart contract code without offering additional life-cycle capabilities, and are thus assigned to the **technology layer (iii)**. Suitable for non-technical users are approaches which abstract blockchain and platform-specific concepts. The modeling language iContractML [14] has a visual notation with few elements for the specification of the structure of smart contracts. Models are translated to DAML, which is compatible with various chains. Based on this language, iContractBot [25] allows the user to specify models conversationally. Another approach targeting multiple chains is the aforementioned Das Contract, in which the behavior of a contract is specified in Blockly. Since Blockly contains coding concepts, we do not consider it suitable for non-technical users. Approaches specifically for Ethereum are LATTE [30] and VersiSolid [23]. The former relies on a combination of form-based definition of the structure of Solidity contracts and their implementation, defined visually in a notation similar to flow-charts. In the latter, Solidity contracts are modeled as state machines in visual fashion. This approach focuses on the formal verification of the generated contract. Another platform-specific approach is UML2Go [20] for Hyperledger Fabric. Contracts are modeled as UML class and sequence diagrams and then translated to Go chaincode using model transformation.

The results show that academic approaches predominantly focus on the platform-specific generation of smart contract code, while holistic solutions are sparse.

---

<sup>2</sup> <https://ontochain.ngi.eu/>

## 5 Industrial Low-Code and No-Code Approaches

For practitioners, an increasing number of low-code and no-code solutions is available. In an informal compilation by Invernizzi and Tossell<sup>3</sup>, solutions from 145 companies were found, such as website and app builders, e-commerce services, and data dashboards. The identified solutions differ substantially in the scope and applications they target. With the aim of assessing the scope and applicability of industrial approaches towards blockchain applications, we conducted a review of state-of-the-art solutions. The data sources for this review are (DS-1): the compilation by Invernizzi and Tossell, (DS-2): practical approaches from prior research [15], and (DS-3): additional research on blockchain-specific no-code and low-code solutions available on the web.

### 5.1 Review Process

We applied a three-step process, consisting of an initial filtering step (S-1), the evaluation of scope and applicability for blockchains in step (S-2), and the classification of solutions applicable to blockchains (S-3). Initially, 169 solutions were identified. In (S-1), we manually retrieved descriptions from the vendor websites in addition to information provided by (DS-1), followed by filtering out duplicate entries, those that could not be reached on the web, or did not provide sufficient information on their websites (e.g. closed beta software). The remaining 150 solutions were evaluated in (S-2) regarding their scope of blockchain integration. Finally, 40 solutions were identified as applicable for blockchains.

### 5.2 Results

For discussing available platforms and their blockchain integration, we distinguish between *1st degree* and *2nd degree* integration. A platform supports 1st degree integration if it interacts directly with blockchains through its software or services. 2nd degree integration is supported if an external service could be integrated that offers 1st degree integration. The criteria for the selected platforms (S-3) listed in Table 3 are that they (a) offer blockchain integration of 1st or 2nd degree and (b) were considered a low-code or no-code approach.

**Categories with 1st degree integration:** 1st degree blockchain integration has been found in 17 solutions intended for building websites and apps, workflow automation, and smart contract development. Exemplary integration features in the *app builder* category are the creation of decentralized apps (Dapps) and the integration of cryptocurrency-related data, e.g. price information. App builders such as Outsystems (5) and Bubble (7) support Dapps, where components of a mobile, desktop, or web app can send blockchain transactions and call smart contract functions, e.g. through the MetaMask browser extension. For *website builders*, blockchain integration has only been found for integrating

<sup>3</sup> <https://pinver.medium.com/decoding-the-no-code-low-code-startup-universe-and-its-players-4b5e0221d58b>



Cat.	Name	Website	d <sub>1</sub>	d <sub>2</sub>	s	Cat.	Name	Website	d <sub>1</sub>	d <sub>2</sub>	s
1	AM Adalo	adalo.com	-	+	o	21	SC DAML	daml.com	+	-	+
2	AM BuildFire	buildfire.com	-	+	o	22	SC Simba Chain	simbachain.com	+	-	-
3	AM Glide	glideapps.com	+	+	-	23	SC Dappbuilder	dappbuilder.io	+	-	+
4	AM Axonator	axonator.com	-	+	-	24	SP Airtable	airtable.com	-	+	-
5	AW Outsystems	outsystems.com	+	-	-	25	WA n8n	n8n.io	+	-	+
6	AW Builder.ai	builder.ai	+	-	-	26	WA Zapier	zapier.com	+	+	o
7	AW Bubble	bubble.io	+	+	-	27	WA Integromat	integromat.com	+	+	-
8	AW Landbot	landbot.io	-	+	-	28	WA Process Str.	process.st	-	+	-
9	AW Draftbit	draftbit.com	-	+	o	29	WA IFTTT	ifttt.com	+	+	-
10	D Parabola	parabola.io	-	+	-	30	WA NodeRed	nodered.org	+	+	+
11	D Gyana	gyana.com	-	+	o	31	WA Aurachain	aurachain.ch	+	-	-
12	D Obviously AI	obviously.ai	-	+	-	32	WB Webflow	webflow.com	+	+	-
13	D Levy	levity.ai	-	+	-	33	WB Unstack	unstack.com	-	+	-
14	F Arengu	arengu.com	-	+	o	34	WB Squarespace	squarespace.com	-	+	-
15	F Formstack	formstack.com	-	+	-	35	WB Linktree	linktr.ee	-	+	-
16	F Tally	tally.so	-	+	-	36	WB Pory	pory.io	-	+	-
17	IN Budibase	budibase.com	-	+	-	37	WB Softr	softr.io	-	+	-
18	IN Flowdash	flowdash.com	-	+	-	38	WB Xooa	xooa.com	+	-	o
19	IN Jet Admin	jetadmin.io	-	+	o	39	WB ICME	icme.io	+	-	-
20	IN Windward	windwardstudios.com	-	+	-	40	WB Atra	atra.io	+	-	o

Cat.: Category, AM: app builder with mobile focus, AW: app builder with web focus, D: data  
F: forms, IN: internal tools, SC: smart contracts, SP: Spreadsheets, WA: workflow automation  
WB: website builders, d<sub>1</sub>: 1st degree integration, d<sub>2</sub>: 2nd degree integration, s: open source  
+: applicable, o: partially applicable, -: not applicable

**Table 3.** Low- and no-code approaches with 1st or 2nd degree blockchain integration.

cryptocurrency-related data, with the exception of ICME (39). ICME is a website builder for creating websites on the Dfinity blockchain. The app and the resulting websites are hosted on Dfinity.

*Workflow automation* tools allow for the execution of user-defined workflows. A workflow is entered via a visual flow-based editor, showing the subsequent flow of steps for execution along with execution logic, or using dialogs or forms. Exemplary integration features are transactions and smart contract support for the Ethereum blockchain in Zapier (26) and Aurachain (31), and support for the Hyperledger Fabric blockchain in NodeRed (30) and Aurachain (31). Further services support crypto-currency data integrations.

*Smart contract development* is supported by integration features in Hyperledger Fabric, Hyperledger Sawtooth, Amazon QLDB, and others in DAML (21), a domain-specific language for textual descriptions of smart contracts. The textual language uses a syntax with natural language elements that can be interpreted and deployed for the supported platforms. Smart contract design based on templates and a visual editor is found for Ethereum, Hyperledger Fabric, and others in SimbaChain (22). The editor supports the creation of smart contracts by defining assets and transactions. Dappbuilder (23) offers smart contract creation from pre-defined templates for Ethereum, Polygon, and others. The approach limits applicability to standardized contracts, e.g. issuing tokens according to the Ethereum ERC-20 and similar token standards.

**Categories with 2nd degree integration:** 2nd degree blockchain integration has been found in 30 solutions intended for building websites, apps, or forms, for workflow automation, internal tools for companies, and for data processing and spreadsheets. 7 solutions also offer 1st degree blockchain integration. The integration features across the categories rely on another services providing a direct integration for blockchain applications. Among the no-code or low-code applications, it is typical to integrate other services in the fashion of a composition, for example, creating an application in an app builder with data provided by an external service. Blockchain integration features, due to this capability, rely on other services for blockchain integration.

Notably, 28 of the 30 solutions integrate with Zapier (26), thereby offering support for *interacting with Ethereum smart contracts and transactions*. These concern website and app builders such as Glide (3), which can embed dialogs for smart contracts and transactions in this way, in addition to integrating cryptocurrency data. Similarly, form builders allow defining input fields and the processing of submitted data through integrations. Arengu (14) is a typical example which also supports visualization with a flow-based editor.

*Workflow automation* tools offer the integration as part of the executable workflow definition. For example, a transaction may be sent after the workflow has been started by another action such as entering data in a spreadsheet. This is often accomplished by integrating AirTable (24). Internal tools include software tools for enterprises, automating typical enterprise resource tasks or operational tasks, e.g. using JetAdmin (19), or business processes as in Flowdash (18). Integrations in this context can be triggered similar to workflow automation tools.

*Data processing and spreadsheets* tools permit integrating data sources, thereby enabling for example the processing of newly appearing blockchain transactions, filtering for specified criteria, and calculations such as the aggregation of transferred amounts. Examples where this is possible are the spreadsheet tool AirTable (24) and data analytics tools such as Parabola (10).

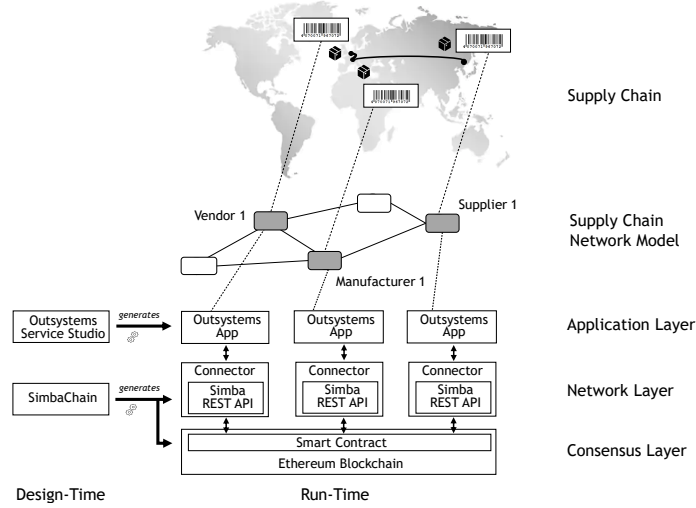
The results show that the integration possibilities for the creation of websites or apps hinge on few services such as Zapier (26), predominantly found in the workflow automation category. Typical integration features consist of access to blockchain transactions or cryptocurrency data. Further integration possibilities with APIs on a technical level are very common, however, they were not considered no-code or low-code when using Webhooks, Rest, other forms of HTTP requests or technical API descriptions. For the development of smart contracts, few no-code instances could be found in practice, with all of them requiring expert knowledge in blockchains.

## 6 Use Case for Low-Code Blockchain Development

For conducting a first evaluation of the state-of-the-art in realizing blockchain applications using low-code and no-code approaches (RQ2), we implemented a blockchain app with a smart contract in the area of supply chain tracking and tracing. For this purpose, we selected the Outsystems low-code platform, which

targets developers, together with the SimbaChain platform as an exemplary no-code platform that is directed towards end-users.

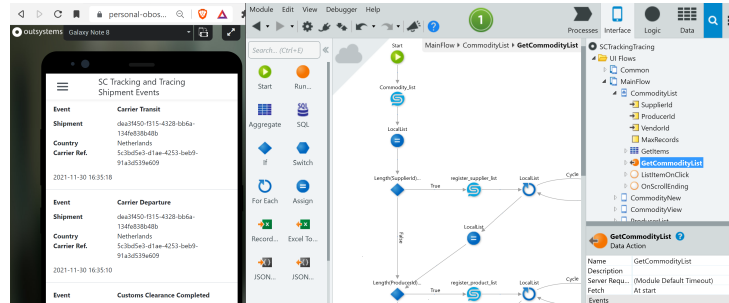
The goal was to provide a trusted and up-to-date IT system shared by distributed supply chain participants. In this domain, blockchain-based solutions promise information that is available as a trusted source in near-time or real-time among network participants [16,7]. In particular, the tracking of goods in international shipments is a challenging area, involving the coordination of material flows from suppliers and manufacturers through container and sea freight companies to distributors. Additionally, products and materials need to be traced back to their source. Without a trusted IT infrastructure, shipments are mostly documented using paper documents, with point-to-point communication by e-mail, phone, and siloed IT systems, resulting in high transaction costs [37,7].



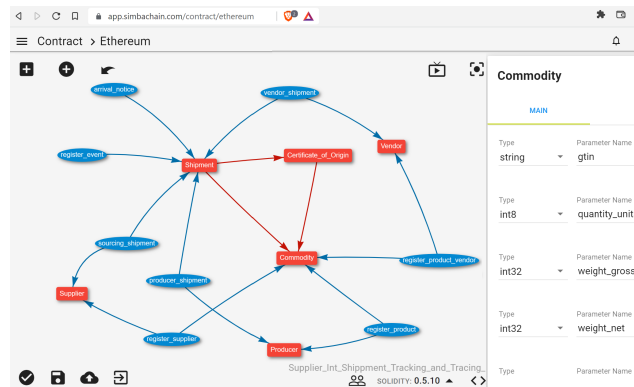
**Fig. 2.** Blockchain-based Architecture for Supply Chain Tracking and Tracing.

Figure 2 shows the implemented three-layer architecture with an exemplary network. Using Outsystems studio, an app was designed for registering suppliers, manufacturers, and vendors together with freight forwarding companies, commodities, and shipments. On the application layer, Supplier 1 might scan a shipment through a Global Trade Item Number (GTIN) with a smartphone camera and submit related IDs and attributes. For Manufacturer 1 and Vendor 1, this data becomes available and is updated with shipment events by freight providers and forwarders. Figure 3 shows this data in the app during development. The Ethereum blockchain is integrated for establishing a consistent view of data on the network and consensus layers of the architecture. In Outsystems, a REST API hosted by Simba relays requests to the Ethereum smart contract. Smart

contracts and APIs are generated together through SimbaChain by specifying the model in Figure 4.



**Fig. 3.** Design of mobile app (left-hand side) in Outsystems Service Studio (right-hand side) using a flow-based editor for processing commodity data records of shipments.



**Fig. 4.** Smart contract design in SimbaChain using a visual editor for a data model of transactions (blue) and assets (red).

## 7 Discussion and Requirements for Future Developments

The review of academic approaches for model-driven engineering for blockchain applications has shown that most approaches focus on the technical level and not all of them support code generation. Rather, many approaches target the formal verification of smart contracts and only some approaches provide working prototypes. Approaches that integrate the business, application, and technical layer have not yet been proposed prominently in the literature. These would

however bring benefits in terms of a holistic view on blockchain application design and should be investigated in the future.

The reviewed no-code and low-code approaches as used in industry showed the high maturity of these platforms. This concerned in particular the high usability, the availability of a broad range of interfaces for cloud-based and blockchain integrations and the possibility of cross-platform development. On the downside, it is hard to trace errors and debug applications on some low-code platforms as implementation details are hidden. Although some platforms offer the inspection of the generated code, this requires again technical know-how.

The practical use case permitted further insights. Regarding the blockchain implementation through SimbaChain, a major architectural limitation is the generation of APIs used as relay when accessing the smart contract. Additional validation of the blockchain is required for assessing the consistency of data. From a user perspective, the SimbaChain platform requires only high-level knowledge of data types in addition to the visual entity concept documented in the platform. While SimbaChain might thus be considered a *no-code* approach, it is limited to the presented operations and its resulting implementation requires expert knowledge for evaluating implementation trade-offs. The app development with Outsystems allows for a visual modeling of program actions and control structures as shown in Figure 3. The specification of the individual elements as well as other application components required the knowledge of software development concepts, such as variables, datatypes, event listeners, and HTTP request and, in one case, debugging through the logging and interception of requests. On the other hand, Outsystems might be considered a *low-code* approach with complex capabilities suitable for developers. During development, code consistency, spotting errors visually, discussing and communicating with domain experts, and cross-platform generation have proven beneficial.

## 8 Conclusion

In this paper we reviewed academic model-driven engineering approaches and industrial low-code and no-code platforms for supporting the development of blockchain-based applications. Whereas academic approaches mostly focus on the technical aspects of development, industrial approaches showed a high maturity in terms of usability and integration capabilities. For future developments, more holistic, cloud-based approaches involving business, application, and technical layers seem desirable. With regard to academic approaches, the provision of integration capabilities and sustainable prototypical implementations present current major challenges.

## Acknowledgment

This work was supported by the Swiss National Science Foundation project Domain-Specific Conceptual Modeling for Distributed Ledger Technologies [196889].

## References

1. Ait Hsain, Y., Laaz, N., Mbarki, S.: Ethereum's smart contracts construction and development using model driven engineering technologies: a review. *Procedia Computer Science* **184**, 785–790 (2021)
2. Antonopoulos, A.M., Wood, G.: *Mastering ethereum: building smart contracts and dapps*. O'reilly Media (2018)
3. Babkin, E., Komleva, N.: Model-driven liaison of organization modeling approaches and blockchain platforms. In: *Advances in Enterprise Engineering XIII*. pp. 167–186. Springer (2020)
4. Bock, A.C., Frank, U.: Low-Code Platform. *Business & Information Systems Engineering* **63**(6), 733–740 (2021)
5. Brambilla, M., Cabot, J., Wimmer, M.: *Model-driven software engineering in practice*, second edition. *Synthesis Lectures on Software Engineering* **3**(1), 1–207 (2017)
6. vom Brocke, J., Simons, A., Riemer, K., Niehaves, B., Plattfaut, R., Clevén, A.: Standing on the shoulders of giants: Challenges and recommendations of literature search in information systems research. *Commun. Assoc. Inf. Syst.* **37**, 9 (2015)
7. Chen, W., Botchie, D., Braganza, A., Han, H.: A transaction cost perspective on blockchain governance in global value chains. *Strategic Change* **31**(1), 75–87 (2022)
8. Clohessy, T., Acton, T., Rogers, N.: *Blockchain Adoption: Technological, Organisational and Environmental Considerations*, pp. 47–76. Springer (2019)
9. Curty, S., Härer, F., Fill, H.G.: Towards the comparison of blockchain-based applications using enterprise modeling. In: *ER Demos/Posters*. CEUR-WS (2021)
10. Di Ruscio, D., Kolovos, D., de Lara, J., Pierantonio, A., Tisi, M., Wimmer, M.: Low-code development and model-driven engineering: Two sides of the same coin? *Software and Systems Modeling* (Jan 2022)
11. Fairley, P.: Ethereum will cut back its absurd energy use. *IEEE Spectrum* **56**(1), 29–32 (2019)
12. Fill, H.G., Härer, F., Muff, F., Curty, S.: Towards augmented enterprise models as low-code interfaces to digital systems. In: *International Symposium on Business Modeling and Software Design*. pp. 343–352. Springer (2021)
13. Foxley, W., Kim, C.: Valid points: Ethereum's proof-of-stake may happen sooner than you think (Mar 2021), <https://www.coindesk.com/tech/2021/03/17/valid-points-ethereums-proof-of-stake-may-happen-sooner-than-you-think/>
14. Hamdaqa, M., Metz, L.A.P., Qasse, I.: IContractML: A domain-specific language for modeling and deploying smart contracts onto multiple blockchain platforms. In: *12th System Analysis and Modelling Conference*. p. 34–43. ACM (2020)
15. Härer, F., Fill, H.G.: A Comparison of Approaches for Visualizing Blockchains and Smart Contracts. *Jusletter IT Weblaw* **February 2019** (2019)
16. Helo, P., Shamsuzzoha, A.: Real-time supply chain—A blockchain architecture for project deliveries. *Robotics and Computer-Integrated Manufacturing* **63**, 101909 (Jun 2020)
17. van den Heuvel, W.J., Tamburri, D.A., D'Amici, D., Izzo, F., Potten, S.: Chainsops for smart contract-based distributed applications. In: *Business Modeling and Software Design*. pp. 374–383. Springer (2021)
18. Holotiuk, F., Moormann, J.: Organizational adoption of digital innovation: the case of blockchain technology. In: *ECIS Conference*. p. 202 (2018)
19. Hu, B., Zhang, Z., Liu, J., Liu, Y., Yin, J., Lu, R., Lin, X.: A comprehensive survey on smart contract construction and execution: paradigms, tools, and systems. *Patterns* **2**(2), 100179 (2021)

20. Jurgelaitis, M., Drungilas, V., Čeponienė, L., Vaičiukynas, E., Butkienė, R., Čeponis, J.: Smart contract code generation from platform specific model for hyperledger go. In: Trends and Applications in Information Systems and Technologies. pp. 63–73. Springer (2021)
21. Levasseur, O., Iqbal, M., Matulevicius, R.: Survey of Model-Driven Engineering Techniques for Blockchain-Based Applications. In: Proceedings of the Forum at Practice of Enterprise Modeling 2021. vol. 3045, pp. 11–20. CEUR (Nov 2021)
22. López-Pintado, O., Dumas, M., García-Bañuelos, L., Weber, I.: Interpreted execution of business process models on blockchain. In: 2019 IEEE 23rd International Enterprise Distributed Object Computing Conference (EDOC). pp. 206–215 (2019)
23. Mavridou, A., Laszka, A., Stachtari, E., Dubey, A.: Verisolid: Correct-by-design smart contracts for ethereum. In: Goldberg, I., Moore, T. (eds.) Financial Cryptography and Data Security. pp. 446–465. Springer (2019)
24. Nguyen, C.T., Hoang, D.T., Nguyen, D.N., Niyato, D., Nguyen, H.T., Dutkiewicz, E.: Proof-of-stake consensus mechanisms for future blockchain networks: Fundamentals, applications and opportunities. *IEEE Access* **7**, 85727–85745 (2019)
25. Qasse, I., Mishra, S., Hamdaqa, M.: iContractBot: A chatbot for smart contracts' specification and code generation. In: IEEE/ACM 3rd Int. Workshop on Bots in Software Engineering. pp. 35–38 (2021)
26. Sahay, A., Indamutsa, A., Di Ruscio, D., Pierantonio, A.: Supporting the understanding and comparison of low-code development platforms. In: SEAA Conference. pp. 171–178. IEEE (2020)
27. Schmidt, D.: Guest editor's introduction: Model-driven engineering. *Computer* **39**(2), 25–31 (2006)
28. Skotnica, M., Pergl, R.: Das contract - a visual domain specific language for modeling blockchain smart contracts. In: Advances in Enterprise Engineering XIII. pp. 149–166. Springer (2020)
29. Sánchez-Gómez, N., Torres-Valderrama, J., García-García, J.A., Gutiérrez, J.J., Escalona, M.J.: Model-based software design and testing in blockchain smart contracts: A systematic literature review. *IEEE Access* **8**, 164556–164569 (2020)
30. Tan, S., S Bhowmick, S., Chua, H.E., Xiao, X.: Latte: Visual construction of smart contracts. In: International Conference on Management of Data. p. 2713–2716. ACM, New York, NY, USA (2020)
31. Tisi, M., Mottu, J.M., Kolovos, D., De Lara, J., Guerra, E., Di Ruscio, D., Pierantonio, A., Wimmer, M.: Lowcomote: Training the next generation of experts in scalable low-code engineering platforms. In: STAF 2019 (2019)
32. Trebbau, S., Wizenty, P., Sachweh, S.: Towards integrating blockchains with microservice architecture using model-driven engineering. In: Agile Processes in Software Engineering and Extreme Programming. pp. 167–175. Springer (2021)
33. Vacca, A., Di Sorbo, A., Visaggio, C.A., Canfora, G.: A systematic literature review of blockchain and smart contract development: Techniques, tools, and open challenges. *Journal of Systems and Software* **174**, 110891 (2021)
34. Watson, R.T., Webster, J.: Analysing the past to prepare for the future: Writing a literature review a roadmap for release 2.0. *J. Decis. Syst.* **29**(3), 129–147 (2020)
35. Webster, J., Watson, R.T.: Analyzing the past to prepare for the future: Writing a literature review. *MIS Q.* **26**(2) (2002)
36. Whittle, J., Hutchinson, J., Rouncefield, M.: The state of practice in model-driven engineering. *IEEE software* **31**(3), 79–85 (2013)
37. Zeng, F., Chan, H.K., Pawar, K.: The adoption of open platform for container bookings in the maritime supply chain. *Transportation Research Part E* **141**(C) (2020)