Preprint from Proc. Intl. Conf. on Service Oriented Computing (ICSOC), 2016

Towards a Shared Ledger Business Collaboration Language based on Data-Aware Processes

Richard Hull[1], Vishal S. Batra[2], Yi-Min Chen[1],
Alin Deutsch[3], Fenno F. Terry Heath, III[1], Victor Vianu[3]

[1]IBM T.J. Watson Research Center
Yorktown Heights, NY, USA
{hull,ymchee,theath}@us.ibm.com

[2]IBM India Research Lab
New Delhi, DL, India
vibatra1@in.ibm.com

[3]University of California, San Diego
La Jolla, CA, USA
{deutsch,vianu}@cs.ucsd.edu

Abstract. Shared ledger technologies, as exemplified by Blockchain, provide a new framework for supporting business collaborations that is based on having a high-reliability, shared, trusted, privacy-preserving, nonrepudiable data repository that includes programmable logic in the form of "smart contracts". The framework has the potential to dramatically transform business collaboration across numerous industry sectors, including finance, supply chain, food production, pharmaceuticals, and healthcare. Widespread adoption of this technology will be accelerated by the development of business-level languages for specifying smart contracts. This paper proposes that data-aware business processes, and in particular the Business Artifact paradigm, can provide a robust basis for a shared ledger Business Collaboration Language (BCL). The fundamental rationale for adopting data-aware processes is that shared ledgers focus on both data and process in equal measure. The paper examines potential advantages of the artifact-based approach from two perspectives: conceptual modeling, and opportunities for formal reasoning (verification). Broad research challenges for the development, understanding, and usage of a shared ledger BCL are highlighted.

1 Introduction

The shared ledger paradigm, as exemplified by Blockchain, was first introduced in Bitcoin [37] to enable a cryptocurrency, but a number of industries are seeing strong potential for generalizations that will dramatically increase the efficiency of many different kinds of business collaboration. A leading initiative towards diverse applications of Blockchain is Hyperledger, a consortium led by the Linux Foundation that includes a global family of partners from finance, banking, Internet of Things, supply chains, manufacturing and Technology [29]; other initiatives include Etherium, R3, and Digital Asset. Widespread adoption of this technology will be accelerated by the development of business-level languages for specifying *smart contracts*, i.e., the programs that run on shared ledgers.

This paper proposes that data-aware business processes [27], and in particular the Business Artifact paradigm [39, 30], can provide a robust basis for a shared ledger *Business Collaboration Language* (BCL).

The core value of Blockchain for business collaboration is that it provides *high-reliability, shared, trusted, privacy-preserving, non-repudiable data repositories*. This is achieved through families of clever, intricate algorithms relating to encryption, distributed computing, and consensus. Data updates to Blockchains obey the "ACID" transactional properties of classical database systems. Furthermore, updates to a blockchain can trigger execution of smart contracts, so blockchains are reminiscent of active databases. Using a blockchain a group of businesses can share data and invoke agreed upon processing in connection with a collaboration in a secure and selective manner. In the absence of Blockchain technology, most multi-party business collaborations are implemented in the form of multiple binary relationships. This leads to more intricate modeling and programming, and increased cost of tracking down the root causes of issues and disagreements. In contrast, blockchains hold the promise of supporting a much more holistic view of business collaborations, and can give immediate transparency to all relevant stakeholders if conflicts arise. They can also simplify the use of analytics to understand collaborations in the aggregate.

We recall the principle of "logical separation" from relational databases, which helps to insulate relational database design and query langauges such as SQL from the physical storage of data on disk. By analogy, we believe that a shared ledger BCL can be designed based on abstractions suitable for business leaders and analysts, and can be largely insulated from Blockchain implementation details. There is early evidence that this is quite feasible: reference [48] shows how the business process language BPMN can be mapped into executable smart contracts on the Ethereum Blockchain. This suggests that a BCL based on various other abstractions can also be successfully mapped onto Blockchain.

A Business Collaboration Language will be a form of *domain specific language*. As such, we expect that developing and maintaining smart contracts with a BCL will be substantially faster and cheaper than using the base smart contract languages on blockchains, e.g., Turing complete languages such as Go or Java. Other domain specific languages for Blockchain include R3's Corda [8] and Digital Asset's DAML [19], which focus on financial transactions, and Ethereum's Solidity [43], whose focus is more general.

Conceptually speaking, the data stored in a blockchain can provide a logical "anchor" for a collaboration between businesses. Indeed, in typical blockchain-enabled collaborations it is assumed that all business-relevant data shared between two or more of the participants will be placed onto and persisted in the blockchain (or an auxiliary data store). This brings a fundamental focus on data that is not traditionally part of process-centric BPM paradigms, such as the BPMN standard including the BPMN constructs for conversations and choreography [7].

The data-centricity of Blockchain suggests that we should base a Business Collaboration Language on the field of *data-aware business processes* [27], that is,

business processes or workflows that incorporate data as a first-class citizen along with process. This data-aware approach was introduced in 2003 by the Business Artifact model [39, 30] (see also [28, 12, 9, 15, 5]), and is also found in modern case management [46, 35], and in Business Objects [42, 41, 31]). The core construct for these approaches focuses on key conceptual entities that progress through business operations. Called *business artifacts* (or *cases* or *business objects*), these entities are modeled using both an *information model* and a *lifecycle model*. A classical example of a business artifact type is the concept of Fedex delivery, not the package itself, but the overall phenomenon, starting with a customer request to ship something, details about the shipping and delivery, and also details about payment. For each such delivery, the corresponding business artifact will hold a growing data set, and the progression of the artifact will follow one of the possible paths in the lifecycle model. When modeling a typical scope of business operations, a handful of interacting business artifact types will be used.

From the perspective of Business Process Management (BPM) and the field of services interoperation (including orchestration and choreography), Blockchain brings unique characteristics. It is logically similar to an orchestrator, in that it can serve as a hub that communicates with each participant in a collaboration. But while orchestrators are typically pro-active and controlling, collaborations might use a blockchain in a more re-active manner. That is, the blockchain might be relatively passive and wait for new updates from participants before producing more data and perhaps alerting other participants of changes. In this sense, a blockchain may act more as a facilitator, similar to the loosely coupled style of choreographies. The Artifact-Centric Services Interoperation (ACSI) approach of [26, 32, 5, 6], developed before the emergence of Blockchain, enables this style of collaboration and provides mechanisms for fine-grained control of data privacy and sharing, and of permissions to make updates or invoke services.

The goal of this paper is to examine the suitability of constructs from business artifacts and ACSI as the foundation for a shared ledger BCL. We focus on the core conceptual abstractions of business artifacts, rather than detailed language design. We consider the viability of using business artifacts as the basis for a BCL from two perspectives: conceptual modeling (Section 4) and support for informal and formal reasoning (Section 5). A brief overview of Blockchain is presented in Section 2 and some research challenges are outlined in Section 6.

2 Short overview of Blockchain: The logical level

There are several introductions to Blockchain and shared ledger technology available (e.g., [29, 20, 48]). We provide here a brief overview that focuses on the logical level rather than on, e.g., encryption and consensus. This logical level provides the basis upon which a BCL will be designed, implemented, and used.

While there are variations, a common set of core elements is shared across most blockchain implementations. There are two classes of logical computational actors that perform processing: *peers* that form a blockchain *network* (i.e., set of computational actors working together to support a given blockchain deploy-

ment), and *participants* that are executing on behalf of the businesses (or other organizations) that are collaborating by using the network. Each participant connects to a single peer, that serves as the connection point between the participant and the network.

In traditional blockchain networks it is assumed that the organizations running the peers have no trust relationship established between them. The encryption, consensus, and other algorithms of blockchain guarantee trusted outcomes in this context. Some recent blockchain initiatives, including Hyperledger [29], are constructed to also enable contexts where groups of businesses have trust relationships that exist outside of the blockchain network (e.g., a consortium of banks, or a large retailer and all of its suppliers and transporters). In such contexts the consensus algorithm may work with a subset of the peers, rather than with all of the peers.

For this paper we are primarily concerned with the *application-level data and processing* in a blockchain network, and largely ignore the additional data and processing used to support encryption, consensus, and the like. At the application level the basic unit of executable code on a blockchain is called a *smart contract* (or a *chaincode*). In the context of business collaborations, it is typical that a smart contract is focused on progressing some type of collaboration (or a part of one) towards completion. A service invocation corresponds to recording a particular step of a collaboration onto the blockchain and potentially computing some additional values and/or generating alerts for interested participants. For example, a smart contract might manage various activities associated with fulfilling a Purchase Order, and one service invocation might focus on an update that says, intuitively, that one line item of the Order has been succefully received.

The basic unit of application-level work on a blockchain is a *transaction*. A transaction is *initiated* by a single participant that sends a service invocation to an identified smart contract running in the blockchain. (Technically, the invocation might be against an already running smart contract instance, or might deploy a new instance.) The service invocation is digitally *signed* by the requestor, which allows tracing of who invoked which transactions. Speaking intuitively, in workflow terms a transaction includes automated processing that results from the service invocation; it will not include inputs or activity by any participant except for the initial service invocation. (Actually, techniques are emerging to enable secure, trusted queries to external sources [49], but this is not considered here.) Once the service invocation is made the blockchain network undergoes a significant amount of processing. The service invocation results in one of two outcomes: (a) the associated transaction becomes *committed*, in which case it is recorded on all (or some pre-determined subset) of the peers, or (b) it is *rejected*, in which case it is essentially removed from all of the peers. Transactions in blockchain follow the "ACID" properties of database transactions.

The processing for a service invocation involves distribution of the invocation to all (or a subset) of the peers, checking for validity of the signature, execution of the code invoked by the service invocation, and reaching consensus amongst

the peers. Depending in part on the number of peers, this can take seconds, 10's of seconds, or in some contexts up to a minute [20].

If the transaction associated with a service invocation is committed then the blockchain may return an *output* to the initiating participant. The transaction may be a *query* that does not modify the application data on the blockchain. Or, the transaction may be an *update* that potentially does modify the data. In the case of updates, the blockchain may also generate *alerts* (with payloads) to other participants about the transaction and/or about values in the blockchain that are related to the transaction.

Traditional blockchain networks incorporate a notion of cryptocurrency payments for participation in the blockchain. For example, Bitcoin and Ethereum have constructs relating to the *transaction fee* to be paid for transaction processing. Other approaches, such as Hyperledger, do not have built-in cryptocurrency, and optionally enable payment for participation through a separate mechanism that is essentially outside of the application-level processing on the blockchain.

In a typical setup, a blockchain network works on multiple initiated transactions in a group. A specific sequencing of the transactions in this group is determined as part of the consensus building process. Once it is determined which of the transactions in the group are to be committed or rejected, and in which order, the application-level data relating to the results of executing the committed transactions is recorded as a *block*. (In some cases, some of the transaction data is recorded into an auxiliary data store rather than in the block itself; see next paragraph.) No semantic relationship is implied about the transactions that are combined into a block. The blockchain itself is essentially a sequential linked list of these blocks. All peers hold an identical copy of this list. (A variation is used for networks in which subsets of peers perform consensus building.)

A blockchain network may maintain a persistent, replicated data store that is referred to by transactions and is updated according to the committed transactions. As one example, Hyperledger maintains a store organized around key-value pairs. In some networks the cost or time involved in storing large volumes of data may be prohibitive. In this case auxiliary stores might be maintained to hold selected data that is encrypted but not widely replicated (see [48]).

In most networks one smart contract can invoke another one. In typical setups, at the logical level the call to the second smart contract is *synchronous*, i.e., the first smart contract will wait until the second smart contract finishes its work and returns a value or a handshake. There may be a family of smart contract invocations stemming from a single service invocation by a participant. All of these invocations are considered to be part of a single transaction; if it is successful at the consensus level then the combined result of the invocations are committed to the blockchain.

It is common to use a "factory" paradigm when working with smart contracts. That is, most smart contracts are written to be used to support numerous instances of collaborations (e.g., financial trades, importing contracts, etc.), where each instance may involve a different set of participants. As the industry's ability to work with blockchains grows, we anticipate that eco-systems

of smart contracts and executing instances of them will emerge. For example, a large manufacturer might maintain two levels of smart contract: "umbrella" contracts that focus on establishing shipping costs, etc., for a year-long period, and "shipping" contracts that focus on supporting individual shipments. Also, different development or standards organizations might create smart contracts for different aspects of large-scale business collaborations (e.g., one might focus on import/export and another on trucking within a country); and a single collaboration might rely on instances of smart contracts that were developed by the different organizations.

While many use cases today involve one or several smart contracts that interact within a single blockchain network, we also anticipate a future with interacting smart contracts that run on different blockchain networks [29]. The messaging between such smart contracts will most likely be asynchronous, because of the time it takes for single blockchain transactions to commit, and the fact that the block commit cycles on different networks will not be aligned.

Blockchain technologies are still evolving, and new features and capabilities will continue to emerge. As such, the selection of abstractions for a BCL should not depend too closely on the capabilities of one blockchain technology. Also, the needs of a BCL may imply the desirability of certain capabilities in the underlying networks, and help to guide how the core blockchain technologies evolve.

3 Overview of Business Artifacts and ACSI approach

This section overviews the business artifact approach for modeling business operations, including the notion of Artifact-Centric Services Interoperation (ACSI). The presentation here is informal and by example; the reader is referred to the numerous articles about business artifacts, including the surveys [12, 28, 9, 15]. We illustrate some of the core notions of business artifacts with an example that could be executed on a shared ledger. The illustration is based on the use of Finite State Machines (FSM's) for the artifact lifecycles, but quite different styles of lifecycle meta-models can also be used (see below).

The modeling focus in business artifacts is on key business-relevant entities that progress through a business, or through a collaboration amongst businesses. Some of the literature uses the phrases *Business Entities* or *Business Entities with Lifecycles* to refer to business artifacts. Typical examples are a Purchase Order or a Financial Transaction; each of these might go through various stages of activity, achieve various business objectives along the way, and access and create data values. In a typical business modeling context there will be a handful of relevant kinds of business entity, that interact in specified ways.

A (*business*) *artifact type* (or schema) consists primarily in an *information model* and a *lifecycle model*. There are numerous options for the meta-model used for the information models, we focus here on nested relations that satisfy the natural property that for each relation (top-level or nested) the scalar columns form a key (see [24, 1]). There are also several options for the meta-model used

for the lifecycle models. Our running example uses FSM's; other options are discussed below. Various mechanisms have been studied for specifying the interaction between business artifacts, including messages [44], service invocations [11], and the ability for conditional triggering to refer to multiple artifacts [25]. In the running example we use service invocations, which parallels the typical style of communication between smart contracts on blockchains.

We now introduce a simplified example that uses business artifacts to support business collaborations in the space of manufacturing commerce. This example loosely follows the running example of [48], and is focused on the management of orders for industrial equipment (e.g., freeze driers for large-scale pharmaceutical packaging), including management of ordering and shipping of the component parts to the manufacturer. In the example there are five kinds of participants: Buyers (who issue Purchase Orders for machines), Manufacturers (who build them), Middlemen (who facilitate purchases and shipments of component parts for the Manufacturers), Suppliers (of component parts), and Shippers.

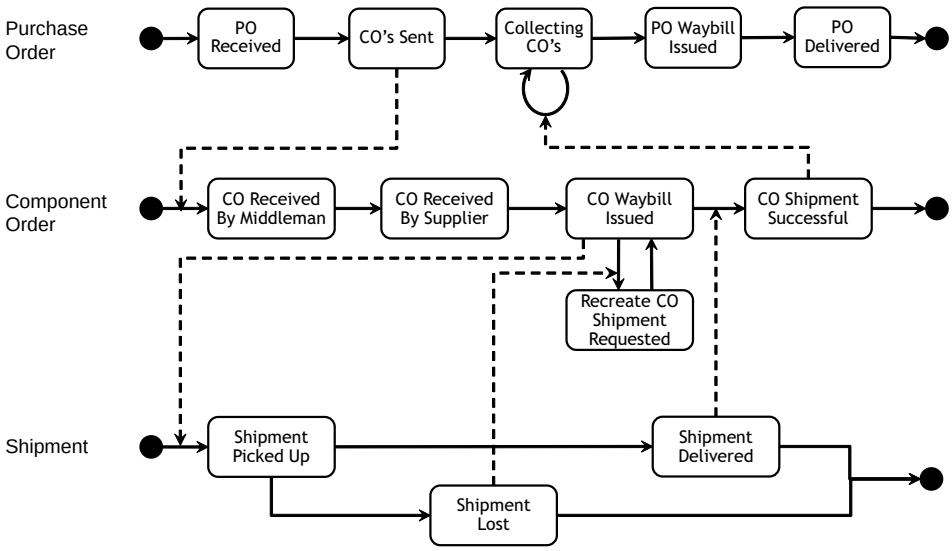

Fig. 1. Sketch of the lifecycle schemas of three interacting business artifact types, including interactions between them (simplified)

Figures 1 and 2 together provide a high-level, simplified illustration of three business artifact types that can support this collaboration. The three artifact types are *Purchase Order* (*PO*), *Component Order* (*CO*), and *Shipment*. The first figure shows sketches of their FSM-based lifecycles (many details have been omitted), and the second one shows a snapshot of the data that might be held by an instance of PO at one point in its progression. Importantly, the artifact types are focused on data and processing that are relevant to the collaboration

itself; the tasks performed individually by the respective participants are out of scope for the model supported on the blockchain.

In a typical execution of artifacts based on these artifact types, a Buyer would issue a purchase order for a machine, leading to the creation of a PO artifact. After analyzing the order, the Manufacturer would determine which component parts are needed and order them from one or more Middlemen. A new CO artifact instance is created for each component part. Focusing on the lifecycle for CO, each FSM state can be viewed intuitively as a *milestone*, i.e., a key business objective that the collaboration around the CO may achieve. The CO's would progress through a lifecycle involving receipt by the Middleman, posting onto the blockchain by the Middleman of a Supplier Order, posting onto the blockchain of a Waybill by the Supplier (e.g., after the Supplier has created the component and chosen a Shipper), and (hopefully) reaching the state where the component has been shipped to the Manufacturer. In this simplified example most exception handling is not included, except for the case of lost shipments.. In particular, if a Shipment moves into the Shipment Lost state this will invoke a service to the CO that another shipment should be created.

PO ID	PO State	Price	Mfr.	Component	CO ID	Middleman	Received Date
PO123	Collecting CO's	$1,500,000	JLH Inc.	Vacuum Pump	CO111	AllPro Solutions	10 July 2016
				Condenser	CO222	Central Logistics	24 June 2016
				Shelving	CO333	AllPro Solutions	–

Fig. 2. Representative snapshot of data held by the information model in an instance of the Purchase Order artifact type (simplified)

Figure 2 shows a simplifed example of the information held by a PO artifact instance. In practice, a PO will typically hold a wealth of information about a PO that is business-relevant to the progression of the collaboration, including dates, contract signatories, relevant history of exceptions and how they were handled, etc. It would also be natural to include a history of service invocations and FSM states visited. The artifact information model provides a conceptually convenient place for storing any information of interest, in business-digestable form. This contrasts with typical BPMN-based solutions, where a lot of business-relevant data may become buried in message bodies and system logs.

The solid lines in Figure 1 correspond to state transitions of the FSM's that result from service invocations. The dashed edges indicate synchronous service invocations that are made between artifact instances. (The synchronous call will result in one state transition in the called FSM, after which control will return to the calling FSM.) Solid edges without incoming dashed edges correspond to services that are invoked from outside the blockchain, i.e., by participants to the collaboration. Some dashed edges have the effect of launching new artifact instances, while others in this simplifed example have the effect of causing state transitions in the called artifact. (In the diagram each dashed line points to

a transition; in practice, a service invocation might cause different transitions depending on the payload of the service invocation and the data held by the called artifact.)

The approach to supporting business collaboration illustrated in this example follows the basic form of Artifact-Centric Services Interoperation (ACSI) hubs as introduced in [26] in 2009. That paper developed access control constructs that seem especially well-suited for use in the shared ledger context. Three forms of access control are defined there. First, a *view* over an artifact type includes restrictions on both the data and the lifecycle. The data portion of a view is essentially a projection of the data (i.e., subset of the columns). For the lifecycle, a view specifies a *condensation* of the FSM, that is, a mapping of the FSM to a new FSM, where multiple states of the source FSM might get mapped to a single state of the target FSM (certain well-formedness restrictions must be followed). This has the effect of hiding from some participants the possible steps that other participants might be involved in. The second form of access control is *window*, which is essentially a selection condition on what artifact instances a participant can see. For example, we would expect that a Shipper can see only the Shipment instances that he is directly involved in supporting. Finally, [26] supports the specification of Create-Read-Update-Delete-Append-Execute (*CRUDAE*) access controls, to limit the ways that participants can modify artifact instances. The BizArtifact system [32, 5, 6] supports variations of the view, window, and CRUDAE constructs of [26] for two artifact lifecycle meta-models. These abstractions are important for a BCL, because they provide intuitive mechanisms for specifying confidentiality constraints in a business-oriented framework. These constructs will also be useful for understanding compatibility of smart contracts for interoperation.

Essentially any process-oriented meta-model can be used for the lifecycle models of business artifacts, including, e.g., Petri nets, BPMN [7], or state charts. For example, the proclets meta-model [47] uses Petri nets to specify a family of interacting processes; similar to business artifacts they provide a top-down way to factor business operations into coherent chunks, although data is not explicitly modeled. Business Objects (e.g., [42, 41, 31]) use FSMs for the lifecycle models and messages for communication between objects. The Guard-Stage-Milestone (GSM) variant of business artifacts [25, 13] provides a declarative, rules-based lifecycle meta-model in which condition-based milestones are supported, and where tasks are hierarchically grouped into *stages* that are launched if certain conditions are met. GSM was used as a basis for the OMG Case Management Model and Notation (CMMN) standard [35]. As discussed in Section 4, the modularity, hierarchy, and declarative characteristics of GSM and CMMN may prove useful when specifying intricate smart contracts at the business level.

4 The Case for Business Artifacts: Conceptual Modeling

Business artifacts were developed to reduce the conceptual distance between (i) how business leaders and analysts think about the processes supporting business

operations, and (ii) how those processes are implemented in practice [39, 12]. This section reviews this and other benefits of the business artifacts and ACSI approaches in the context of supporting business collaborations on shared ledgers. The section includes comparison with process-centric approaches to model collaborations and choreographies.

Holistic, top-down factoring based on business-relevant concepts. The central tenet of business artifacts modeling of business operations is to focus on the key business entities, including milestones they may achieve and the data needed and/or produced to achieve them. These business entities are often the conceptual building blocks used by business leaders to think about their operations, and the milestones provide the basis for many of the Key Performance Indicators (KPI's) that the operations are measured by [33]. Additional information that naturally fits into a business artifact information model, such as the price paid, production cost, time taken, etc., provides the basis for additional KPI's, some of which relate directly to a business's financial performance.

An effective business modeling method. The Business Entity Lifecycle Analysis (BELA) method [44] was developed at IBM to support business operations modeling using business artifacts. The approach follows a five-stage process (with some back-and-forth) as follows: (1) Identify the key business artifact types in the business scope; (2) Identify key milesteons for these artifact types and place these into a sequencing diagram (e.g., an FSM or something more declarative); (3) Identify the data needed and/or produced to achieve these milestones; (4) Identify the tasks needed to achieve the milestones; and (5) Identify interactions between the artifacts. The BELA method has been applied in numerous application areas (e.g., [3]), and is supported in the IBM Service-Oriented Method and Architecture (SOMA) tool suite [38].

In multiple situations the BELA method (and its precursors) solved business modeling and deployment challenges that the Lean Six Sigma approach was unable to effectively resolve [3, 10]. The BELA method was found most effective in two kinds of contexts: (a) where the targeted business operations spanned across multiple business silos, and (b) where multiple organizations (e.g., obtained through acquisitions) were performing essentially the same function for different geographic areas. Context (a) is of course very relevant to Blockchain-enabled collaborations.

Modeling flexible processes. Business artifacts information models provide a unified store of business-relevant data, which enables natural, declarative, rules-based approaches such as GSM and CMMN for specifying process. This in turn enables artifacts to model rich flexibility in processes. Consider for example the use case of mortgage origination, that is, the processing involved in obtaining a mortgage loan for a house. In this process a broad variety of documents and data is gathered from numerous stakeholders, and then evaluated in various ways by a small set of stakeholders (e.g., bank, underwriter, insurance company). The documents and data arrive in various orders, processing can be done on subsets of the incoming data, and sometimes new versions of the documents or data are needed. A rules-based lifecycle specification enables flexible response to the

data as it arrives. Combining a rules-based style with hierarchy further increases modeling flexibility. For example, it makes it easier to support variation as might arise when working on a given use case across differently-sized contracts, with various partners, and involving different countries.

Another kind of flexibility relates to distribution of responsibilities between participants. Business artifacts enable a separation of concerns, with business-relevant data and milestones at one level, and distribution across participants at a second level. This enables the use of a single family of business artifact types to support instances of business collaboration where responsibilities are divided among participants in differing ways.

Support for evolution. It has been argued that the business artifact [3, 44] and Business Objects [42, 41, 31] approaches enable evolution of process models in ways that are both intuitively natural and relatively inexpensive. This again stems from the top-down factoring of models created using these approaches. This flexibility will be useful as business needs change and new variations on existing smart contracts are created.

Comparison with process-centric approaches. It is informative to compare and contrast the use of business artifacts and ACSI to support business collaboration vis-a-vis process-centric approaches. We focus here on the embodiment found in the BPMN version 2.0 [7] frameworks for *collaboration* and *choreography*. Reference [48] mentioned above provides an illustration of a BPMN collaboration and the corresponding BPMN choreography.

BPMN *collaboration* focuses on the parties in a collaboration, where the internal process of each party is represented in a separate BPMN pool (that is, a specification of a BPMN process, possibly spread across swimlanes that are performed by different participants and/or roles). Messages between those pools (in particular, between *send tasks* and *receive tasks*) guide the collaborative process. BPMN *choreography* enables a complimentary view. The primary building block is the *choreography activity*, which in turn may be a *choreography task*, a subchoreography, or a "call choreography" which acts as a placeholder for a choreography. A choreography task corresponds to an interaction between two or more partipants, where one of these is the initiator. Choreography activities are strung together using flow constructs (including gateways for conditionals, joins, etc.)

In both BMPN collaboration and choreography the information shared between businesses is modeled using a family of individual messages that go between susbsets of participants. The ACSI approach provides a paradigm shift, because the information shared between businesses is modeled in a single, logically coherent data store. This data store is organized around business artifacts that can hold all business-relevant information about the collaboration, regardless of which participants create or use it. Data privacy is layered on top. This paradigm shift, from modeling shared information in messages to modeling it in a top-down unified way, can provide substantial benefits in design, implementation, maintenance, reporting on progress, and tracking of disputes and exceptions. As noted above, the use of a unified data model permits the use

of rules-based lifecycle models, which in turn enables intuitive specification of highly flexible kinds of collaborations.

The artifact-centric approach provides a natural, flexible way to represent 1-many relationships between the conceptual entites. The example of Section 3 illustrates a 1-many relationship between PO's and CO's; more complex 1-many relationships can arise in shipping scenarios where shipments and invoices may refer to overlapping sets of line items, or in financial applications where, e.g, mortgage loans are bundled into mortgage-backed securities, grouped into tranches (e.g., groupings based on differing risk levels), and then divided amongst multiple buyers. BPMN processes support *multi-instance* tasks and sub-processes, BPM collaborations support *multi-instance* pools, and BPMN collaborations support *multi-instance* choreography activities. All of these use constructs that essentially package the multi-instance aspect into a sub-process of the parent instance. As a result, it may be very cumbersome in BPMN to faithfully represent some styles of business artifact interactions, e.g., where a parent artifact progresses through several states, and in parallel the child artifacts progress through their states, with interleaved interactions between parent and children. In contrast, rules-based artifact lifecycle models can support such interactions in an intuitive and succinct manner.

Potential for community adoption. Case management has emerged as an important style of Business Process Management, especially in connection with knowledge-worker intensive processes. The number of case management deployments continues to grow, and so does the number of case management savvy business analysts and developers. The underlying paradigms of case management and business artifacts are very close [35]. Business analysts and developers will be able to bring their experiences and knowledge of case management to the creation of artifact-centric smart contracts.

Turning now to FSM-based artifacts in particular, we mention a report from the U.S. Office of Financial Research [21], that argues that many financial contracts can and should be represented using FSM's. The paper encourages workers in the financial field, who have had little or no exposure to programming or Computer Science abstractions, to make the effort to understand how FSM's can bring value and systematic, repeatable approaches to financial exchanges.

Academic foundations. The business artifact and related approaches have spawned a broad and growing body of academic and industrial research since the first publications in 2003, as indicated in the surveys [28, 9].

5 The Case for Business Artifacts: Formal Reasoning

The ability to reason about smart contracts, including their interaction with each other and with external business processes, will be a crucial enabler in the success and wide-spread adoption of the shared ledger approach. Both informal and formal styles of reasoning will be significant. By "informal reasoning" we mean the kind of reasoning that developers and others often carry out to convince themselves that programs will operate as desired. By "formal reaoning" we mean

both mathematical styles of reasoning (such as a proof of that two-phase locking ensures serializability of transactions) and automated reasoning (as in automatic verification by tools such as SPIN [23] or WAVE [17]). Smart contracts involve both data and process at fundamental levels, and so reasoning about process, reasoning about data, and reasoning about process together with data will all bring important value.

Effective support of informal reasoning will rely in part on conceptual models that are natural and intuitive for both business analysts and developers, and that enable focus on critical aspects of the processing, including achievement of key business goals, smart contract interactions, and privacy guarantees. The discussion in Section 4 suggests several reasons why the business artifact approach can provide an appropriate basis for this informal reasoning.

The use of FSM's for artifact lifecycles can bring strong advantages, because of their intuitive simplicity and the wealth of widely known informal intuitions and formal algorithms and results about them. However, reasoning about data will be essential, given the importance of ensuring data privacy across different participants. Data can be incorporated in varying degrees, e.g., as follows.

(a) Ignore data, and furthermore focus on situations where there is a bounded number of artifact instances (e.g., at most 10 CO's per PO).
(b) Include modeling of 1-many relationships between artifact instances, e.g., between a PO and its CO's. In one variation, the number of CO's may be unbounded and essentially unrestricted. In another variation, the number of CO's might be unbounded in general, but for each PO there might be a bound on the number of CO's based on the initial input.
(c) Ability to specify structural properties on the data held by one or more artifact instances. This might include referential constraints, e.g., for each instance of Shipment there must be a CO that refers to it, and in turn the ID of that CO is held in the PO data. It might also include key and functional dependencies, perhaps extended to the nested relation context, e.g., that across all CO instances, each shipper can be associated with only one middleman.
(d) It might include arithmetic properties, e.g., that the price of a PO must be below \$5M. The arithmetic properties might cut across artifact instances and rely on aggregation, e.g., that the total price of all CO's is below some percentage of the overall price of the PO.
(e) Finally, these might include *privacy ensuring* constraints. An informal example is that no middleman can see the price associated with any CO that he is not handling.

We now turn more specifically to automatic verification. This relies on precisely defined abstractions that capture key elements of a framework. This sometimes brings a loss of completeness in the model being studied (e.g., by ignoring data, or arithmetical relationships between data values). In the context of automatic verification, achieving decidability and relatively low complexity in the presence of data can require carefully designed restrictions on the model studied. It is typical for verifiers to support algorithms that are *sound* (they are never

wrong when they claims correctness of a specification) but not necessarily *complete* (they may produce false negatives, i.e., candidate counter-examples to the desired property, which need to be validated by the user). Importantly, the perspectives and tools developed from the formal perspective may also be helpful for the informal reasoning. As just one example, formal languages developed to specify integrity constraints (both static and temporal) may help in the formulation of targets for informal reasoning, and also lead to design principles. The theory of relational database integrity constraints illustrates how these kinds of mathematical models can have far-reaching practical impact.

A primary goal of formal reasoning about smart contracts is to ensure that during operation they will achieve certain goals, and they will avoid various conditions. The conditions to avoid may be specified in terms of *static constraints*. In the running example, e.g., we may want to avoid any situation where there is a CO and a corresponding Shipment, where the CO is in the state Shipment Delivered and the Shipment is in state Recreate Shipment Requested. This can be thought of as a constraint on a cross-product of all the FSM's of the relevant artifact instances. Richer static constraints involving data might be considered, e.g., relating to structural properties and (aggregate) arithmetic properties.

For such constraints, formal reasoning will typically focus on *reachability*, that is, given (i) a static constraint and (ii) a class of possible input sequences, determine if there is any input sequence that leads to a family of artifact instances that violates the constraint. If data is ignored (and the number of artifact instances is bounded) this reduces essentially to reachability in FSMs. In some contexts this has complexity NLOGSPACE-complete and is tractable. In more general settings, e.g., if the FSM's have non-determinism (as might arise when data is abstracted away), or with the richer kinds of temporal constraints discussed below, verification is PSPACE-complete. Approaches have been developed (e.g., state vector models [22]) to enable practical algorithms for such verification problems. The modeling approach of Petri Nets provides another broad family of verification results to draw upon for contexts where data is ignored [18].

Reachability of states violating a static constraint is in fact a form of *temporal constraint*. More broadly, it is common to reason about the behaviors of a system by using temporal operators, such as those from Linear Temporal Logic (LTL) [40]: **G** (always), **F** (eventually), **X** (next), and **U** (until). For example, $\mathbf{G}p$ (where p is a propositional variable) says that p holds at all times in the run, $\mathbf{F}p$ says that p will eventually hold, and $\mathbf{G}(p \rightarrow \mathbf{F}q)$ says that whenever p holds, q must hold sometime in the future. While LTL (and its variants) are useful for FSM's and other models without data, an extension is needed to include data-aware properties, such as items (b) through (e) in the list given above. To this end an extension of LTL is used, called here LTL-FO [34], in which First-Order (FO) logic formulas are used in place of propositional variables in LTL formulas.

Verification problems for smart contracts with data can be formulated along the lines originally developed in [4, 14] for business artifacts. In that setting, each (parameterized) invocable action (a.k.a., "service") is associated with (parameterized) pre- and post-conditions expressed in FO. A common restriction is

that the pre- and post-conditions be expressed in existential FO (that is, only existential quantifiers). Suppose now that we are given a system SC with one or more smart contracts, which includes a family $\mathcal{A}$ of invocable actions. Suppose further that φ is an LTL-FO formula. The basic verification problem for SC and φ can be stated as follows: Does every run of SC satisfy φ?

There is already a rich family of results for automatic verification of data-aware processes [15, 9]. Several restricted classes of artifact systems have been studied, along with temporal properties expressed in restricted variants of LTL-FO. This includes classes that impose restrictions on how the actions (services) manipulate set-valued attributes, and classes that permit orderings on domain elements, permit arithmetic operators, etc. In most cases the verification problem for the restricted family has worst-case complexity of PSPACE-complete, which is reasonable given that the classical propositional model-checking problem is PSPACE-complete. Further, the WAVE verification tool [17] provides evidence that verification for data-aware processes can be practical. Also, recent theoretical work leverages hierarchy in GSM artifact specifications to enable optimizations in verification algorithms [16].

6 Conclusions

This paper has discussed the merit of using abstractions from business artifacts and its relatives as the basis for a shared ledger Business Collaboration Language (BCL). The use of these abstractions for Blockchain brings several research challenges, some of which are highlighted below.

Modeling abstractions. There are several variations on the basic theme of business artifacts, based mainly a spectrum of possible lifecycle meta-models, ranging from the fully procedural to the fully declarative. A key challenge is choosing the right mix of abstractions that cover the use cases from multiple industry sectors. Designing a coherent BCL that supports the more procedural style of, e.g., FSM's, and also the more declarative style of, e.g., GSM and CMMN, is a particular challenge.

Views An important aspect of the meta-model underlying a BCL will be support for intuitive ways to specify access rights, that is, to ensure privacy of data and processing steps. Another important area concerns specifying interfaces for how business artifacts from one smart contract will interact with business artifacts from smart contracts. Some notion of business artifact *view* and related constructs (e.g., [26]) will be central in helping to make progress in these areas; see also [2, 36].

"On-ramps" to Blockchain: If Blockchain is successful in providing a new level of efficiency for business collaborations, then many businesses will need to convert existing collaborating business processes into blockchain-enabled ones. Adapting legacy business processes will bring many challenges, including the impedance mismatch between the data-aware style of coordination enabled by Blockchain vs. the process-centric style of most legacy processes. The BELA

method [44] for business process design can be adapted to help with the conversions, and [45] may provide a useful starting point for managing linkages between data on a blockchain and data maintained by collaboration participants.

Reasoning and Validation. A variety of tools are needed to give strong confidence in the smart contracts that people deploy and mix together. This will include well thought out testing strategies and frameworks, and also succinct representations to simplify human reasoning about the contracts. As suggested in Section 5, automatic verification techniques and results for business artifacts can be extended to the Blockchain context. This brings new questions to the forefront, including how to obtain practical verification results in connection with 1-many relationships between business artifacts.

Implementation. Realizing the vision of Blockchain, including scalability and reasonable running speeds, is an on-going work. Finding accurate and efficient ways to support a BCL on top of Blockchain adds another dimension to the engineering challenges. The use of a BCL may permit certain optimizations, analogous to optimizations of database query languages. It remains open whether it will be better to implement a BCL using auto code generation, i.e., mapping a BCL program into, e.g., a Go or Java smart contract, or to create a BCL interpreter that can run on top of blockchains.

Smart Contract Eco-systems. In the coming years we anticipate large libraries of smart contracts that are designed to interact. In a BCL each smart contract might be based on one or several business artifact types. Tools are needed for discovering smart contracts based on artifact types and their milestones, for quickly checking compatibility between them, and for reasoning about and testing behaviors resulting from their interactions.

References

1. S. Abiteboul and N. Bidoit. Non first normal form relations: An algebra allowing data restructuring. *J. Comput. Syst. Sci.*, 33(3):361–393, 1986.
2. S. Abiteboul and V. Vianu. Collaborative data-driven workflows: think global, act local. In *Intl. Symp. Principles of Database Systems (PODS)*, 2013.
3. K. Bhattacharya, N. S. Caswell, S. Kumaran, A. Nigam, and F. Y. Wu. Artifact-centered operational modeling: Lessons from customer engagements. *IBM Sys. J.*, 46(4):703–721, 2007.
4. K. Bhattacharya, C. Gerede, R. Hull, R. Liu, and J. Su. Towards formal analysis of artifact-centric business process models. In *Intl. Conf. Business Process Mgmt. (BPM)*, pages 288–304, 2007.
5. D. Boaz, T. Heath, M. Gupta, L. Limonad, Y. Sun, R. Hull, and R. Vaculín. The ACSI hub: A data-centric environment for service interoperation. In *Proc. BPM Demo Sessions*, 2014.
6. D. Boaz, L. Limonad, and M. Gupta. BizArtifact: Artifact-centric Business Process Management (open-source code base), June 2013. `http://sourceforge.net/projects/bizartifact/`, accessed 20 July 2016.
7. Business Process Model and Notation (BPMN), version 2.0, 3 January 2011. `http://www.omg.org/spec/BPMN/2.0`, accessed 10 July 2016.

8. R. G. Brown. Introducing R3 Corda^{TM}: A Distributed Ledger Designed for Financial Services. `http://r3cev.com/blog/2016/4/4/introducing-r3-corda-a-distributed-ledger-designed-for-financial-services`, accessed 20 July 2016.
9. D. Calvanese, G. De Giacomo, and M. Montali. Foundations of data-aware process analysis: a database theory perspective. In *Intl. Symp. Principles of Database Systems, (PODS)*, 2013.
10. T. Chao et al. Artifact-based transformation of IBM Global Financing: A case study. In *Intl. Conf. on Business Process Mgmt. (BPM)*, 2009.
11. D. Cohn, P. Dhoolia, F. T. Heath III, F. Pinel, and J. Vergo. Siena: From powerpoint to web app in 5 minutes. In *Intl. Conf. Services Oriented Computing (ICSOC)*, 2008.
12. D. Cohn and R. Hull. Business artifacts: A data-centric approach to modeling business operations and processes. *IEEE Data Eng. Bull.*, 32:3–9, 2009.
13. E. Damaggio, R. Hull, and R. Vaculín. On the equivalence of incremental and fixpoint semantics for business artifacts with guard-stage-milestone lifecycles. *Information Systems*, 38:561–584, 2013.
14. A. Deutsch, R. Hull, F. Patrizi, and V. Vianu. Automatic verification of data-centric business processes. In *Intl. Conf. on Database Theory (ICDT)*, 2009.
15. A. Deutsch, R. Hull, and V. Vianu. Automatic verification of database-centric systems. *SIGMOD Record*, 43(3):5–17, 2014.
16. A. Deutsch, Y. Li, and V. Vianu. Verification of hierarchical artifact systems. In *Intl. Symp. on Principles of Database Systems, (PODS)*, 2016.
17. A. Deutsch and V. Vianu. WAVE: Automatic verification of data-driven web services. *IEEE Data Eng. Bull.*, 31(3):35–39, 2008.
18. M. Diaz, editor. *Petri Nets: Fundamental Models, Verification and Applications*. Wiley, Jersey City, NJ, USA, 2009.
19. Introducing the Digital Asset Modeling Language. `https://digitalasset.com/press/introducing-daml.html`, accessed 20 July 2016.
20. A Next-Generation Smart Contract and Decentralized Application Platform, 2016. `https://github.com/ethereum/wiki/wiki/White-Paper`, accessed 20 July 2016.
21. M. D. Flood and O. R. Goodenough. Contract as automaton: The computational representation of financial agreements, 26 March 2015. `https://financialresearch.gov/working-papers/files/OFRwp-2015-04_Contract-as-Automaton-The-Computational-Representation-of-Financial-Agreements.pdf`, accessed 16 July 2016.
22. G. J. Holzmann. An improved protocol reachability analysis technique. *Softw., Pract. Exper.*, 18(2):137–161, 1988.
23. G. J. Holzmann and D. Bosnacki. Multi-core model checking with SPIN. In *Intl. Parallel and Distributed Processing Symposium (IPDPS)*, 2007.
24. G. Hulin. On restructuring nested relations in partitioned normal form. In *Intl. Conf. on Very Large Data Bases (VLDB)*, 1990.
25. R. Hull et al. Business artifacts with guard-stage-milestone lifecycles: Managing artifact interactions with conditions and events. In *ACM Intl. Conf. on Distributed Event-based Systems (DEBS)*, 2011.
26. R. Hull, N. Narendra, and A. Nigam. Facilitating workflow interoperation using artifact-centric hubs. In *Intl. Conf. Service Oriented Computing (ICSOC)*, 2009.
27. R. Hull and J. Su. Report on NSF Workshop on Data-Centric Workflows. http://dcw2009.cs.ucsb.edu/report.pdf, 2012.
28. R. Hull, J. Su, and R. Vaculín. Data management perspectives on business process management: tutorial overview. In *Intl. Conf. on Mgmt. of Data (SIGMOD)*, 2013.

29. Hyperledger whitepaper, 2015. www.the-blockchain.com/docs/Hyperledger%20Whitepaper.pdf, accessed 16 July 2016.
30. S. Kumaran, P. Nandi, F. T. Heath III, K. Bhaskaran, and R. Das. ADoc-oriented programming. In *Symp. on Applications and the Internet (SAINT)*, 2003.
31. V. Künzle and M. Reichert. PHILharmonicflows: towards a framework for object-aware process management. *Journal of Software Maintenance*, 23(4):205–244, 2011.
32. L. Limonad, D. Boaz, R. Hull, R. Vaculín, and F. T. Heath III. A generic business artifacts based authorization framework for cross-enterprise collaboration. In *SRII Global Conference*, 2012.
33. R. Liu, R. Vaculín, Z. Shan, A. Nigam, and F. Y. Wu. Business artifact-centric modeling for real-time performance monitoring. In *Intl. Conf. Business Process Mgmt. (BPM)*, 2011.
34. Z. Manna and A. Pnueli. Verification of Concurrent Programs, Part I: The Temporal Framework. Technical Report STAN-CS-81-836, Stanford University, 1981.
35. M. Marin, R. Hull, and R. Vaculín. Data-centric BPM and the emerging Case Management standard: A short survey. In *Business Process Mgmt. Workshops*, 2012.
36. V. Z. Moffitt, J. Stoyanovich, S. Abiteboul, and G. Miklau. Collaborative access control in webdamlog. In *ACM Intl. Conf. Mgmt. of Data (SIGMOD)*, 2015.
37. S. Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System, 2009. https://bitcoin.org/bitcoin.pdf, accessed 16 July 2016.
38. T. Nguyen and M. Fiammante. Match processes to business needs: Apply BELA to case management, October, 2011. http://www-01.ibm.com/software/solutions/soa/newsletter/october11/bela_case_management.html, accessed 20 July 2016.
39. A. Nigam and N. S. Caswell. Business Artifacts: An Approach to Operational Specification. *IBM Systems Journal*, 42(3), 2003.
40. A. Pnueli. The temporal logic of programs. In *Symp. Foundations of Computer Science (FOCS)*, 1977.
41. G. Redding, , et al. A flexible, object-centric approach for business process modelling. *Service Oriented Computing and Applications*, 4(3):191–201, 2010.
42. G. Redding, M. Dumas, A. H. M. ter Hofstede, and A. Iordachescu. Transforming object-oriented models to process-oriented models. In *Business Process Mgmt. Workshops*, volume 4928. Springer, 2007.
43. Solidity. https://solidity.readthedocs.io/en/latest/, accessed 20 July 2016.
44. J. Strosnider, P. Nandi, S. Kumaran, S. Ghosh, and A. Arsanjani. Model-driven synthesis of SOA solutions. *IBM Systems Journal*, 47(3):415–432, 2008.
45. Y. Sun, J. Su, and J. Yang. Universal artifacts: A new approach to business process management (BPM) systems. *ACM Trans. Management Inf. Syst.*, 7(1), 2016.
46. W. van der Aalst, M. Weske, and D. Grünbauer. Case handling: A new paradigm for business process support. *Data Knowl. Eng.*, 53(2):129–162, 2005.
47. W. M. P. van der Aalst, P. Barthelmess, C. Ellis, and J. Wainer. Proclets: A framework for lightweight interacting workflow processes. *Int. J. Coop. Inf. Syst.*, 10(4):443–481, 2001.
48. I. Weber, X. S. Xu, R. Riveret, G. Governatori, A. Ponomarev, and J. Mendling. Untrusted business process monitoring and execution using blockchain. In *Intl. Conf. Business Process Mgmt. (BPM)*, Rio de Janeiro, Brazil, Sept. 2016.
49. F. Zhang, E. Cecchetti, K. Croman, A. Juels, and E. Shi. Town crier: An authenticated data feed for smart contracts. *IACR Cryptology ePrint Archive*, 2016, 2016. http://eprint.iacr.org/2016/168, accessed 20 July 2016.