Documenting the Execution of Semantically Modelled Inter-organisational Workflows on a Distributed Ledger

Christoph H.-J. Braun
Institute AIFB
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
braun@kit.edu

Janina Traue
Blockchain@DATEV
DATEV eG
Nuremberg, Germany
janina.traue@datev.de

Boris Lingl
Blockchain@DATEV
DATEV eG
Nuremberg, Germany
boris.lingl@datev.de

Tobias Käfer
Institute AIFB
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
tobias.kaefer@kit.edu

***Abstract*—We present an approach for documenting the execution of inter-organisational workflows on a distributed ledger, with the possibility of adding selectively shared verifiable data to the workflow instances' documentation. On data level, we base our approach on semantic workflow and data modelling, augmented with hashing. As system components, we use a distributed ledger as consensus-based shared database for workflow documentation and data verification, and RESTful APIs for sharing data with different privacy and publicity requirements. We evaluate our approach using a load test based in a real-world logistics setting.**

***Index Terms*—Distributed ledger, Decentralised semantic data management, workflow management**

I. Introduction

Specialisation, globalisation, and vertical disintegration of companies lead to value chains and associated workflows that are (1) highly inter-organisational and (2) where often not all parties throughout the value chain (i.e. workflow) are defined prior to instance execution time[1]. The collaboration in such value chains is enabled by information technology and software systems. Yet, heterogenity of the IT system landscape, especially if there are small organisations involved poses the challenge of interoperability throughout the value chain network. In addition, documentation and certification requirements demand verifiable data about the workflows' instances to be shared with all or some members of the network[2]. That is, while information about the workflow needs to be available to all members of the network, other data needs to remain private, under the souvereignty of the owning party, with the possibility to share it with selected parties.

Consider for instance a small or medium enterprise (SME) in manufacturing, which is a typical client of most tax advisors. Such an SME typically has to procure 30 % of the components of the products it sells up-stream on the value chain and is subject to documentation duties. Thus, multiple and changing suppliers and service providers take part in the manufacturing workflow: Assume the SME offers a cleaning product and needs to document its production steps. The production process consists of several steps, from the cleaning fluid itself, which has pre-products from third parties, to the production of bottles, for which there are multiple suppliers, labelling, shipping, handling etc. The challenge for the SME is that its information flow typically terminates at the end of their Enterprise Resource Planning (ERP) solution. Thus, inter-organisational workflows with changing participants are commonplace for SMEs, but hard to support with information technology for verifiable documentation.

The challenge when building a solution is in the combination of a number of technologies that allow for (a) verification in a setting with potentially low trust, (b) interoperability, (c) coordinating the course of action with variable participants, and (d) data souvereignty. Technologies that facilitate those points include: (a) distributed ledgers, (b) semantic descriptions, (c) workflow management solutions with late binding, and (d) decentralised data Pods with RESTful interfaces and fine-granular access control, as put forward by SoLiD[3].

Previous approaches that combine technologies to address only some of the mentioned challenges, for instance: inter-organisational workflows have been investigated, e. g. in [10] using the distributed ledger to drive the workflow execution, which puts a considerable amount of load on the distributed ledger and all workflow-relevant data is public on the distributed ledger. Verifiable, but souvereign data sharing has been investigated in [4] using Linked Data and distributed ledgers, but lacks the workflow aspects. Semantic descriptions for workflows have been investigated e. g. in the WiLD approach [7], but lack late-binding and instance representations on the distributed ledger. We provide a more in-depth discussion of related work in section II, where we also present the subleties in different ways of bringing workflows to distributed ledgers.

Our approach consists in:

[1] https://www.cisco.com/c/dam/en/us/solutions/collateral/enterprise/benefit-unified-communications/c11-593971-00_creating_collab_bus_procWP.pdf

[2] https://ec.europa.eu/docsroom/documents/37824/attachments/2/translations/en/renditions/native

[3] https://solidproject.org/

1) A representation for workflows and instances in the WiLD ontology on the distributed ledger to allow for maintaining workflow instances in WiLD on the distributed ledger
2) An extension to WiLD to allow for late binding to cater for the ad-hoc nature of workflow instances
3) A model of chain events to document the workflow execution and the data to be verified
4) An architecture based on SoLiD and Ethereum

Where 1.-3. are the core contributions of this paper. The architecture is shared with [3, 4].

The paper is structured as follows: In section II, we survey related work. In section III, we give the technological and formal foundations for our approach. In section IV, we present our approach. In section V, we present a performance evaluation. Last, in section VI, we discuss the practical applicability our approach and conclude.

II. Related Work

We work in the intersection of Linked Data, workflows, and distributed ledger. Thus, we survey related work in each intersection.

Linked Data and Workflows

Many ontologies have been defined to describe workflows, including OWL-S [12]. We base our work on the WiLD ontology (workflows in Linked Data), using which one can describe workflows in a way that allows for execution on the web architecture and under the open-world assumption [7].

Linked Data and Distributed Ledger

Third et al. present a vision of LinkChains [21] to combine SoLiD (for personal data storage), distributed ledger (for verification), and IPFS (a distributed peer-to-peer filesystem – for meta data) for verifiable tokens that can be attached to people. Our approach does not use IPFS but makes the participants responsible for maintaining their data over time. Ramachandran et al. describe different approaches for distributing the verification and data management aspect between Linked Data and the distributed ledger [16]. Similarly to [4], Our approach can be categorised in *Storing individual hashes of SoLiD Pod-stored files on the Blockchain*. In [4], while working partially with the same technologies, Braun et al. did not investigate workflow aspects. Sopek et al. investigated an approach to store RDF on the distributed ledger [18], while we store data off-chain and use a specialised Smart Contract to maintain only workflow models and instances on-chain.

Workflows and Distributed Ledger

Different approaches in the intersection of the BPM community, which is concerned with workflow management, use the distributed ledger differently. They can be distinguished along: Who is driving the workflow execution: Is the distributed ledger actively issuing calls to other system components or is the distributed ledger merely passively monitoring? How is the workflow maintained on the distributed ledger: Does the distributed ledger (a) have one or multiple Smart Contracts that interpret a description of a workflow model, or (b) execute Smart Contracts into which the workflow model has been compiled?

Weber et al. [24] present two approaches they call *mediator* and *choreography monitor*. The mediator approach is in our terminology the active approach, the choreography monitor the passive approach. Moreover, they follow the compiled approach. Similar to our approach, they support the passive mode. In contrast to our apporach, they follow the compiled apporach. Another difference is the workflow representation, where they use the BPMN choreography diagram, which focusses on messages exchanged, whereas we use a workflow representation that focusses on the order of activities.

In subsequent work around the Caterpillar approach, see eg. [10], López-Pintado et al. present a full workflow engine based on BPMN, with a wide array of BPMN constructs supported including swimlanes and subprocesses. Their motivation is similar to ours: to enable interorganisational workflows by combining workflow management and distributed ledgers. Caterpillar in [10] is active and follows the compiled style. For this active execution, they also need to put all execution-relevant data onto the distributed ledger. Later, they changed their approach to follow the interpreted style citing improved flexibility and efficiency [9]. In another extension, they extended Caterpillar to support role binding at run-time [8]. Role-binding is also part of our late binding, however in our case, also the workflow model to be bound has to be determined.

The LEAN approach by Sturm et al. [19] shares our motivation when it comes to documentation and traceability. In contrast to the original Caterpillar approach and similar to our apporach, they follow an interpreted style. Yet, while they use one generic Smart Contract that can interpret all workflow models, they use one Smart Contract per workflow instance. In contast, we use one Smart Contract for all workflow models and all workflow instances.

The inclined reader can find a more in-depth discussion of blockchain and workflow management in a paper by Mendling et al. [13], which, next to including the approaches discussed above, also discusses future challenges and opportunities.

III. Preliminaries

In this section, we introduce the technologies, practices, and standards on which we base our approach.

A. Linked Data and SoLiD

The Linked Data principles for data publishing on the Web have been coined by Berners-Lee [2]. They recommend the use of Uniform Resource Indentifiers (URIs) to identify things, the Hypertext Transfer Protocol (HTTP) to transfer data, and the Resource Description Framework (RDF) to describe data, combined with the imperative to provide hyperlinks. Under the headline of Social Linked Data (SoLiD), for an early demonstration see [11], the practices around Linked Data are used and extended to decentrally publish personal data on the web using possibly self-hosted personal online data

storages (Pods) with access control[4] and with the possibility to interact with people via Linked Data Notifications[5]. If people (or more general: agents) are identified using URIs in the SoLiD context, the identifiers are called WebIDs[6]. User-facing applications that consume SoLiD Pods are called SoLiD Apps.

B. Uniform Resource Identifier(URI)

Uniform Resource Identifiers (URIs) serve as names on the web[7]. URIs are meant to identify anything: abstract, virtual, or physical things. URIs are character sequences that start with a so-called scheme followed by a colon character. The scheme determines how the subsequent characters are to be interpreted. The scheme http: or https: denotes that interaction with the resource via the HTTP protocol may be possible.

C. Hypertext Transfer Protocol (HTTP)

The Hypertext Transfer Protocol (HTTP) is the protocol for data transfer on the web[8]. HTTP subdivides the two parties involved in a peer-to-peer communication act into clients, who initiate the communication request and are called user agents in HTTP, and servers, who react to the request with a response and are called origin server in HTTP. The body of an HTTP message (i. e. request or response) can contain data, which can come in different data models and formats. One such format is RDF.

D. Resource Description Framework (RDF)

The Resource Description Framework (RDF)[9] is the graph-based data model of the Linked Data family of technologies. An RDF graph describes the adjacency list of a labelled graph, and is defined as a set of triples. The positions in a triple are called *subject*, *predicate*, and *object*, where in each positions a URI may appear. In subject and object position also graph-local identifiers, so-called blank nodes, are allowed. To describe values, literals can be used in object position. Literals, blank nodes, and URIs are called Terms in RDF. Ontologies are a set of terms with a formally defined meaning. To determine hashes for RDF graphs, the approach by Hogan can be used [6].

E. Workflows in Linked Data (WiLD)

Workflow models are an abstract description of a course of action as a set of activities with a prescibed order. Workflow instances are a concrete incarnation of a workflow model. The WiLD ontology [7] provides terms to describe both, workflow models and instances. Activities in WiLD can be atomic (i. e. no further subdivision is expected), ordered (sequential or parallel) or conditional. Popular workflow languages such as BPMN use a flow-based description of workflows, i. e. the course of action is described step-wise. WiLD in contrast uses a tree-based description of workflows, which can be losslessly converted in to a flow-based representation and vice versa [23]. In the tree, the leaf nodes are the activities from the flow-based diagram, and the upper-level nodes provide structure and bear the control flow information. See Figure 2 for an example with both a tree- and a flow-based representation of a workflow.

F. Distributed Ledger, Smart Contracts, and Ethereum

A distributed ledger is a shared distributed database comprised of a decentralised network of database nodes, where each node holds an identical copy of all data. All data is transparently available on all database nodes. There exists no central authority governing the shared database. New data is added to the database in an append-only style and synchronized among database nodes following a pre-defined consensus protocol. The consensus protocol in place determines the rules by which the nodes accept or reject data proposed for addition to the database. When appending new data, timestamps and hash-based references to previous data leads are included thereby imposing high effort on retrospective modification of data. Taking transparency through data replication into account, a high degree of data integrity is achieved. The first implementation of such system is the Bitcoin blockchain [14].

For our work, we choose Ethereum [5], a well-established blockchain implementation, that allows for the deployment Smart Contracts. Smart Contracts allow for defining application logic that is executed directly on the distributed ledger [20]. The application logic is automatically executed when conditions regarding ledger data as specified in the contract are met. Smart Contracts can be regarded as database-level code whose scope is limited to the data available on the ledger and whose logic can be triggered by issuing specific database transactions.

IV. Approach

We present our approach as follows: first, we present the architecture of our approach to give an high-level-overview. Then, we have a look at the data modelling: we briefly describe how we extend the WiLD ontology to allow for late binding. Subsequently, we describe the on-chain representation of a workflow model into which a Smart Contract code reads RDF describing a workflow model in WiLD. After the data modelling, we describe the dynamics of the system: we present the events that are logged on the chain in response to progress in workflow instance enactment.

A. Architecture

The architecture, illustrated in Figure 1, consists in: a SoLiD Pod for each participant, a distributed ledger with wallets for the participants and a Smart Contract that implements the behaviour outlined in sections IV-C and IV-D, as well as internal service endpoints of the participants and a SoLiD App as a user interface, which both interact with the SoLiD Pods and the distributed ledger. For our implementation, we use Ethereum as distributed ledger, but our approach should also be transferrable to other distributed ledger implementations.

[4]https://solid.github.io/web-access-control-spec/
[5]https://www.w3.org/TR/ldn/
[6]https://www.w3.org/2005/Incubator/webid/spec/
[7]https://tools.ietf.org/html/rfc3986
[8]https://tools.ietf.org/html/rfc7230
[9]https://www.w3.org/TR/rdf11-concepts/

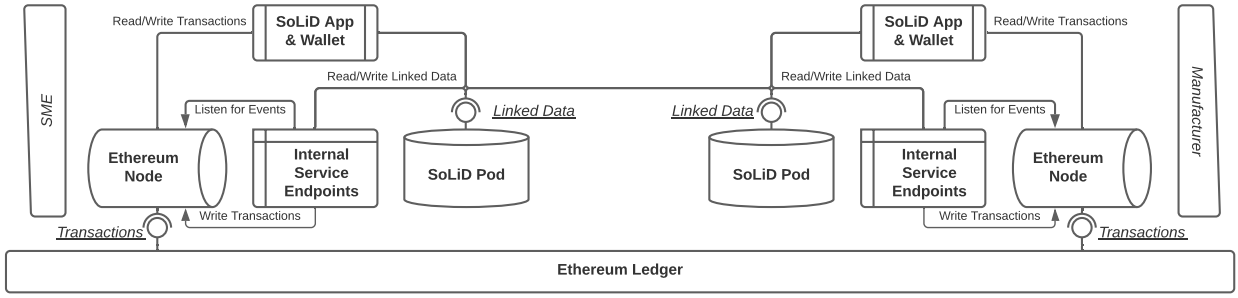

Figure 1. The system architecture: Each user in the network has its own Ethereum Node, Internal Service Endpoints and SoLiD Pod.

On their SoLiD Pod, participants store semantic descriptions of (i) their WebID and profile (linked to their Ethereum wallet address), (ii) workflow models they offer, and (iii) business-relevant data. The participants can SoLiD-based apply access control to e. g. only share workflow models and business-relevant data with selected parties.

On the distributed ledger, each participant has a wallet. The ledger also runs a Smart Contract that offers the functional interface to:

- *Represent* workflow models in WiLD on-chain
- *Document* progress of workflow instances using events
- *Late-bind* additional participants and subprocesses during workflow execution
- *Maintain and verify* hashes of selectively shared data that is stored off-chain in access-controlled SoLiD Pods

In our apporach, there is one single Smart Contract for all workflow models and instances. As this Smart Contract manages all workflow models, we can enable the plugging together of different workflow models by different participants.

The internal service endpoints of a participant provide an interface for coupling existing IT infrastructure for business processes with the distributed ledger and SoLiD Pods. Internal service endpoints consume events documented by the Smart Contract as a trigger for executing their corresponding service. The specific character of the internal service endpoints is dependant on the IT landscape of the corresponding participant.

The SoLiD App provides a user interface to manually interact with data stored on the SoLiD Pods and the Smart Contract on the distributed ledger. The app allows for creating workflow models using the WiLD ontology. A a graphical workflow model editor is offered for convenience. The models can be stored on the distributed ledger and the SoLiD Pod of the participant. In addition, the app allows for manually interaction with the Smart Contract: manual instantiation of workflows, documentaion of workflow activities of workflow instances and late binding of additional participants and subprocesses.

B. Extending WiLD for Late Binding

The WiLD ontology allows for representing workflows in a tree, where leaf nodes are atomic activities, i. e. activities where no further subdivision is foreseen by the modeller. To allow for late binding during workflow execution, we need to introduce a special kind of activity that, similar to atomic activities, only appears on leaf level at modelling time. However, during workflow execution, activities of this kind need to be instantiated with a fresh workflow instance of another workflow. We call this kind of activity a wild:InterfaceActivity, as such activities serve as interface between organisations. A similar notion in BPMN is the 'call activity', which allows for modular re-use of workflow models, mainly within one workflow management system. In the tree representation, the late binding adds a sub-tree to a leaf node, see Figure 2. Late binding has been studied in the context of Web Services. In the classification of Pautasso et al. [15], we do *Binding at Invocation Time*, i. e. at a comparatively late point in time during workflow execution, which allows to delay the decision in long-running workflows. Yet, as the binding is a change that happens before the course of action reaches the activity, we consider the binding as a safe change in the workflow model, for unsafe changes see for instance the investigations of van der Aalst [22].

An example can be found in Figure 2, where workflow models and actors for #production_interface and #transport_ interface have to be dermined at run-time.

C. On-chain Representation

On the chain, the Smart Contract offers the functional interface to re-build the tree-based WiLD representation of workflows by interpreting the RDF triples provided by the function caller: All activities in the tree and their interrelations are represented as sets, mappings, lists, and structs.

In the tree, an activity is characterised by its type and a list of children. We identify each activity on the chain using the activity's URI as a string value. If the activity's list of children is empty, the activity is either an atomic activity or an interface activity. In the latter case, the activity maintains a mapping from workflow instances to workflow instances (see below) in order to point from the containing instance, which provides the interface, to the contained instance. Moreover, we maintain the address of the wallet of the owner of the activity for rights management. Thus, an activity is a pentuple:

$$Activity = (URI, Wallet, ActivityType, [Activity], \\ Instance \rightarrow Instance)$$

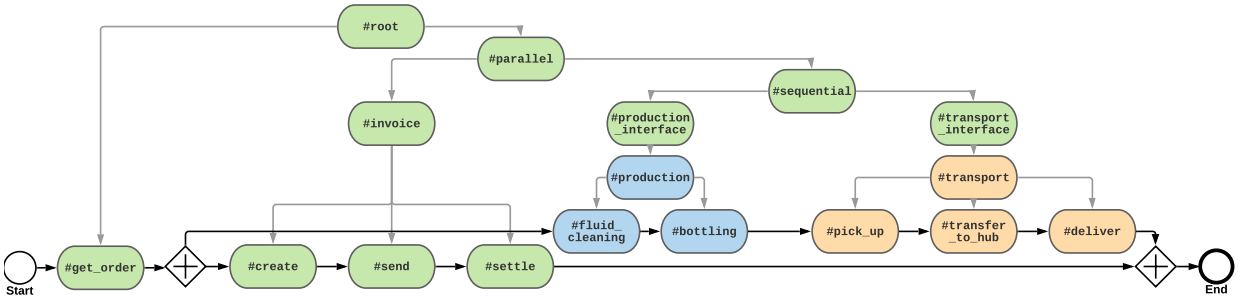

Figure 2. A tree-based workflow in WiLD and its flow-based BPMN counterpart (solid black symbols). Blue and orange: different actors determined at run-time to fill the interfaces.

where the last element of the pentuple is a mapping, the pen-ultimate element is a possibly empty list, and $ActivityType \in \{Atomic, Conditional, Parallel, Sequential, Interface\}$.

A workflow model has a distinguished activity, which is the root in the tree representation of the model. We also identify each workflow on the chain using a string with its URI. We also maintain the owner's wallet for each workflow, again for rights management. On top, we store a hash of the off-chain workflow model in RDF available at the given URI, from which the on-chain representation has been compiled, for the purpose of verification. Thus, a workflow model is a quadruple:

$$WorkflowModel = (URI, Wallet, hash, Activity)$$

A workflow instance refers to the workflow model to which it is an instance, and maintains a mapping from all the activities in the workflow to their state in the instance. As our approach is only concerned with workflow instances that exist on the chain, in constrast to the workflow models, there is no off-chain representation of the instances. Therefore, we can identify the workflow instances by an integer not a URI, which allows us to save some space. We thus describe workflow instances as a triple:

$$Instance = (ID, WorkflowModel, Activity \rightarrow State)$$

where $State \in \{initialised, active, done\}$, and again the last element is a mapping.

D. Evolving the On-chain Representation

The on-chain representation of workflow instances can be evolved by calling the functions of our Smart Contract. The Smart Contract then changes its internal data structures and logs the following events to the distributed ledger:

Instantiation of a Workflow Model: As workflow models are identified by a URI both on-chain and off-chain, the Smart Contract must be supplied with the URI of the workflow model to be instantiated. Next to this URI, the caller of the Smart Contract can supply another URI and the hash of the data to be retrieved at said URI, in order to allow for later verification of the data. The Smart Contract then logs on the chain: the ID of the newly generated workflow instance, next to the workflow model's URI, the other URI and the hash.

Documenting Workflow Instance Progress: The workflow execution is performed outside of the chain. Our Smart Contract allows for documenting this progress with the ID of the workflow instance at hand, the URI of the activity, the new state of the activity in the instance. On top, the caller can supply a URI where progress-relevant data can be obtained and a hash of this data for later verification. Only the workflow owner can change the corresponding instance's activity states.

Offer for a Tender: The interface activity in workflow models allows for plugging in other workflows that do not need to be known at modelling time, but can be assigned at run-time. This late binding allows e. g. for the choice of a range of suppliers for the same service. To 'fill' an interface activity of a specific workflow instance, the agent owning the workflow first determines a suitable workflow, e.g. by searching available workflows on the SoLiD Pods of their business partners. Then, she creates an offer for a tender by issuing a corresponding transaction to the Smart Contract. The offer must contain the URI of the interface activity, the URI of the activity to fill the interface, and the workflow instance ID for which the offer is provided. At the same time, a URI and a hash can be provided, e. g. offer documentation stored in a SoLiD Pod.

Acceptance of an Offer for a Tender: The owner of the workflow requested to fill the interface activity can to accept or reject the offer. According to the owner's decision, the Smart Contract logs on the chain: the URI of the interface activity, whether the offer has been accepted and the workflow instance ID in question. Again, at the same time, a URI and a hash can be provided with more information and for later verification.

The plugged-in workflow extends the existing workflow instance and thus evolves just as its superordinate workflow instance with the interface activity it filled. Activities of the plugged-in workflow remain under control of their owner; the owner of the superordinate workflow is not able to change their states.

V. Evaluation

We evaluate our approach as follows: We take a workflow based on industrial practice, run it in our lab environment, and measure the performance.

As environment, we use 16 cores (Intel Xeon e5-2690 v4 2.6 GHz) of an HP ProLiant DL380 Gen9 server. From those

16 cores, 4 cores, 16 GB RAM, and 32 GB HDD are reserved for a virtual machine running the distributed ledger. Another 4 cores, 12 GB RAM and 16 GB HDD are assigned as virtual machines each to three components that simulate participants in workflows. All virtual machines run Ubuntu Linux. The first virutal machine hosts a proof-of-authority Ethereum network comprised of a signer node and a peer. Specifically, we use the 'Clique' consensus protocol for signing transactions, which has been shown by Angelis et al. to have a comparatively small communication overhead [1]. The remaining three virtual machines each host a system consisting in two components: (1) a SoLiD Pod that serves the workflows one participant offers, next to (2) Java code that (a) listens on the distributed ledger for workflow instantiation events, (b) retrieves the respective workflow from Linked Data, (c) sends transactions to the chain to set activity instances to active/done, as prescribed in the workflow but skipping the actual task, (d) fills interface activities.

The workflow is inspired from e-commerce cases as observed by the project partners of the authors in industrial practice in B2C and B2B settings in Germany. The workflow employs multiple partners for shipping and processing of physical goods.

The evaluation then specifically works as follows: We initialise the evaluation by putting workflow models onto the distributed ledger and the SoLiD Pods. The load using which we evaluate are then instances of those workflow models. We measure the number of transactions on the distributed ledger that our system can perform per second. Thus, a transaction in our case would be for instance a change in state of an activity in an instance, or an accepted tender. From our industrial partners, a peak load of 300 transactions per second would be required to cover a pre-pandemic e-commerce peak load as typically observed during Christmas season.

A. Scale-Up

Our set-up only contains one signer node and one peer node, where the peer node acts as an gateway for receiving transactions from the simulated participants. One would be inclined to run an evaluation that scales the number of nodes. Yet, Schäffer et al. investigated the scalability of different Ethereum blockchains [17] and found no significant performance difference when scaling up the network's size of proof-of-authority networks.

B. Varying Load Ingestion

To see how different dynamics of the load affect our system, we vary the way the load is ingested. We fix the overall load at 1200 instances and we contrast three scenarios:

- **Peak** All instances are ingested at once.
- **Batch** 3 ingestion batches with 20 s delay.
- **Mini** 12 ingestion mini-batches with 5 s delay.

The results can be found in Figure 3. In the Peak scenario, we can see how the different workflow instances are processed in a fairly parallel fashion: The tender and responses (green and blue respectively) are within a discernible period of blocks.

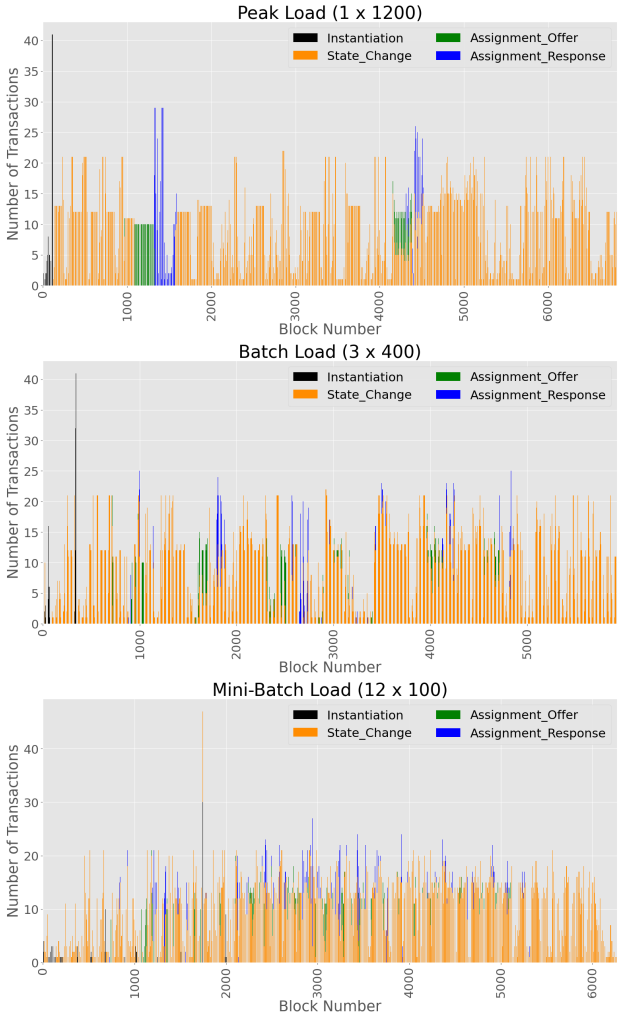

Figure 3. Evaluating our approach using different ingestions of the load.

However in the Batch scenarios, the events of the different types are highly interleaved due to the iterative ingest of instances. Thus, the continuous ingestion of workflow instances is reflected in the overall more leveled firing of events. In the Peak scenario, the average number of transactions per second of around 500 is higher than in the other scenarios (Batch: $\sim$ 375, Mini-Batch: $\sim$ 350) and well above the required level of 300. The smaller we choose the batch size however and move to a more realistic scenario, the more the performance degrades, but is still within a security margin above the required level.

Investigating the reasons of the degradation we can identify the management of nonces, i. e. consecutive transaction numbers, as a cause. Our implementation is multi-threaded, which can lead to race conditions on this consecutive number. We tried two implementations of nonce management:

- **Polite** The party who wants to send a transaction asks the distributed ledger for the next nonce.
- **Eager** The party who wants to send a transaction maintains the number of the nonce that it last tried, increments

if it fails and tries again.

Our implementation of the polite approach rendered the system irresponsive: the many requests of our multi-threaded implementation overloaded our system, because if nonces are requested for multiple queued transactions simultaneously, the same nonce is retrieved as response and all but one transaction are reverted and have to be requeued. We generated Figure 3 using an implementation of the eager approach. Note that the more the access to the ledger is shared between the instatiating part of our implementation and the workflow execution part, the more the performance degrades, but stays well above the acceptance criterion of the industry partner.

VI. Conclusion

In this paper, we presented an approach that combines semantic modelling of data for interoperability, SoLiD as means for data souvereignty, distributed ledger for verification and transparency, and workflow modelling for documenting courses of action. We evaluated our approach in a logistics setting regarding performance. Using our proof-of-concept implementation, we show that with research-quality code and under lab environments, we can provide a research solution that satisfies the requirements of real-world customers and meet their performance requirements.

We envision demand from industry for approaches like ours: Businesses are increasingly connected by value chains, where regulatory demands like the German supply chain act initiative[10] will generate new challenges for companies that call for solutions as presented in this paper. Moreover, recent standardisation efforts in Germany and France make a standardised description format for PDF-based orders appear on the horizon[11], after the corresponding format for invoices has been standardised already. This standardisation builds on older standards including the XML format for Cross-Industry Order (CIO) by the UN/CEFACT[12], for which ERP and manufacturing systems already have been digitised in the past.

In the future, we hope that decentralised identities such as SSIs[13], DIDs[14] (based on semantic technologies), and WebIDs (contained in SoLiD) permeate organisations such that e. g. people and machines can get identified across organisations and ecosystems, where the underlying open standards create an interoperable substrate for, e g. inter-organisational data sharing around supply chains –using solutions like this paper.

Acknowledgments

This work is partially supported by the German federal ministry for education and research (BMBF) in TraPS, a Software Campus project (FKZ 01IS17042).

[10] https://www.bmas.de/SharedDocs/Downloads/DE/Gesetze/Regierungsentwuerfe/reg-sorgfaltspflichtengesetz.pdf

[11] https://www.ferd-net.de/aktuelles/meldungen/order-x-ein-gemeinsamer-standard-fuer-elektronische-bestellungen-in-deutschland-und-frankreich.html

[12] https://unece.org/trade/uncefact

[13] http://www.lifewithalacrity.com/2016/04/the-path-to-self-soverereign-identity.html

[14] http://www.w3.org/TR/did-core/

References

[1] Angelis, SD., Aniello, L., Baldoni, R., Lombardi, F., Margheri, A., and Sassone, V., "PBFT vs Proof-of-Authority: Applying the CAP Theorem to Permissioned Blockchain", Proc. of the 2nd ITASEC, 2018.

[2] Berners-Lee, T., "Linked Data", Design Issues, 2006. https://www.w3.org/DesignIssues/LinkedData.

[3] Braun, C., and Käfer, T., "A SoLiD App to Participate in a Scalable Semantic Supply Chain Network on the Blockchain (Demo)", Proc. of Demos at the 19th ISWC, 2020.

[4] Braun, C., and Käfer, T., "Verifying the Integrity of Hyperlinked Information Using Linked Data and Smart Contracts", Proc. of the 15th SEMANTiCS, 2019.

[5] Buterin, V., "Ethereum white paper", 2013. https://ethereum.org/en/whitepaper/.

[6] Hogan, A., "Skolemising Blank Nodes while Preserving Isomorphism", Proc. of the 24th WWW, 2015.

[7] Käfer, T., and Harth, A., "Specifying, Monitoring, and Executing Workflows in Linked Data Environments", Proc. of the 17th ISWC, 2018.

[8] López-Pintado, O., Dumas, M., García-Bañuelos, L., and Weber, I., "Dynamic Role Binding in Blockchain-Based Collaborative Business Processes", Proc. of the 31st CAiSE, 2019.

[9] López-Pintado, O., Dumas, M., García-Bañuelos, L., and Weber, I., "Interpreted Execution of Business Process Models on Blockchain", Proc. of the 23rd EDOC, 2019.

[10] López-Pintado, O., García-Bañuelos, L., Dumas, M., Weber, I., and Ponomarev, A., "Caterpillar: A business process execution engine on the Ethereum blockchain", Software: Practice and Experience 49(7) 2019.

[11] Mansour, E., "A Demonstration of the Solid Platform for Social Web Applications", Proc. of P&D at the 25th WWW, 2016.

[12] Martin, DL., "Bringing Semantics to Web Services with OWL-S", World Wide Web Journal 2007.

[13] Mendling, J., "Blockchains for Business Process Management - Challenges and Opportunities", CoRR abs/1704.03610 2017.

[14] Nakamoto, S., "Bitcoin: A Peer-to-Peer Electronic Cash System", Cryptography Mailing list at https://metzdowd.com 2009.

[15] Pautasso, C., and Alonso, G., "Flexible Binding for Reusable Composition of Web Services", Proc. of the Software Composition WS at the 4th ETAPS, 2005.

[16] Ramachandran, M., Chowdhury, N., Third, A., Domingue, J., Quick, K., and Bachler, M., "Towards Complete Decentralised Verification of Data with Confidentiality: Different ways to connect Solid Pods and Blockchain", Proc. of DecentralisedWeb WS at the 29th WebConf, 2020.

[17] Schäffer, M., Angelo, MD., and Salzer, G., "Performance and Scalability of Private Ethereum Blockchains", Proc. of the Blockchain and CEE Forum at the BPM, Springer 2019.

[18] Sopek, M., Gradzki, P., Kosowski, W., Kuzinski, D., Trójczak, R., and Trypuz, R., "GraphChain: A Distributed Database with Explicit Semantics and Chained RDF Graphs", Proc. of the LD-DL WS at the 27th WebConf, 2018.

[19] Sturm, C., Szalanczi, J., Schönig, S., and Jablonski, S., "A Lean Architecture for Blockchain Based Decentralized Process Execution", Proc. of the 1st CCBPM workshop at the 16th BPM conference, 2018.

[20] Szabo, N., "Smart Contracts", 1994. http://szabo.best.vwh.net/smart.contracts.html. Offline, but available in the Web Archive.

[21] Third, A., and Domingue, J., "LinkChains: Trusted Personal Linked Data", Proc. of the BlockSW at the 18th ISWC, 2020.

[22] Van der Aalst, WMP., "Exterminating the Dynamic Change Bug: A Concrete Approach to Support Workflow Change", Information Systems Frontiers 3(3) 2001.

[23] Vanhatalo, J., Völzer, H., and Koehler, J., "The Refined Process Structure Tree", Proc. of the 6th BPM, 2008.

[24] Weber, I., Xu, X., Riveret, R., Governatori, G., Ponomarev, A., and Mendling, J., "Untrusted Business Process Monitoring and Execution Using Blockchain", Proc. of the 14th BPM, Springer 2016.