Decentralized Control: A Novel Form of Interorganizational Workflow Interoperability

Christian Sturm, Jonas Szalanczi, Stefan Jablonski, Stefan Schönig

Decentralized Control: A Novel Form of Interorganizational Workflow Interoperability

Christian Sturm[1], Jonas Szalanczi[2], Stefan Jablonski[1], and Stefan Schönig[3]

[1] University of Bayreuth, Bayreuth, Germany
firstname.lastname@uni-bayreuth.de
[2] NeuroForge GmbH & Co. KG
lastname@neuroforge.de
[3] University of Regensburg, Regensburg, Germany
firstname.lastname@ur.de

Abstract. Companies deploy workflow management systems which interpret process models to ensure compliance of operational duties by automatically distribute work items to employees or machinery. Enterprises are often forced to outsource services and join interorganizational collaborations, e.g. in supply chain scenarios, to remain competitive on the market. Today, interorganizational process management cater for interconnecting publicly visible tasks of the participants' local workflows whereby a predefined message exchange protocol ensures interoperability. Especially in flexible large-scale collaboration scenarios, this strategy lacks in global monitoring or transparent data-based routing the control flow due to a missing central controlling instance or global accessible data. Blockchain technology promises to automatically run applications in a decentralized fashion without any trusted third party needed, as participants will agree on a common valid state algorithmically. We capitalize on this consensus finding mechanism and tamper-resistant data storage to propose a novel form of workflow interoperability for interorganizational workflows which autonomously orchestrates the *process in-between*. The approach solves issues regarding the monitoring of the global state, the distribution of work items to the respective business partners and achieves data-based routing by holding in-process variables decentralized in a trustworthy fashion.

Keywords: Process Execution · Choreography · Interorganizational Process Management · Workflow Interoperability · Data-based Routing · Blockchain

1 Introduction

Business Process Management (BPM) is the discipline of modelling, executing and analyzing business processes [5]. The skeleton of a process [10] comprises the set of activities to perform and their temporal order (functional and behavioural perspective), the person, group or machinery which is responsible for the execution (organizational and operational perspective) and related data values (informational perspective) which may affect the other perspectives, for instance

in the way that the control-flow is adapted based on current data values, called *data-based routing*.

BPM distinguishes intraorganizational processes from interorganizational processes. In intraorganizational settings, the organization itself is process owner having all responsibilities and special interest in successfully completing the process. Technically, all data and information remain inside a certain environment, e.g. the companies' boundaries. The process owner implements software that orchestrates internal routines and distributes work items to employees. In interorganizational settings, at least two separate participants or process owners strive to reach one common business goal by outsourcing or distributing the workload. This is accompanied with additional issues like the preservation of company secrets (hide internal workflows) [13], the lack of unconditional trust between participants [27] or a missing common IT infrastructure [19].

Six different possibilities to establish interorganizational workflows are described in literature as forms of interoperability, but each of them comes with certain drawbacks, e.g. prevents parallel execution or lacks monitoring of the global workflow progress [26]. We propose *decentralized control* as a novel form of interoperability to address requirements to interorganizational processes, and to solve the following dilemma: On the one hand, interorganizational processes do not have a particular process owner and a centralized coordinating instance is not considered applicable, so that participants are obliged to hold all data locally. On the other hand, all participants have to agree on a single source of truth to achieve a commonly accepted work item distribution or data-based routing. Decentralized control is build upon a decentralized peer-to-peer network and achieves both, decentralization resp. local data retention and common agreement on the global state of the process flow. To counteract intended fraud attempts or unintended technical malfunctions with locally corrupted data, decentralized control includes algorithms called consensus mechanisms that are applied to establish a fault tolerant system and to eliminate trust issues. In consequence, the group of business partners agree on a single source of truth, which is responsible for the global monitoring and routes the process flow automatically based on the process model. We will use blockchain technology to implement decentralized control in this paper. The Ethereum blockchain is able to store data in a decentralized fashion and with small programs (*smart contracts*) process information can be encapsulated and decisions are taken automatically.

In the remainder of the paper, we describe decentralized control and enlarge upon both modelling and implementation issues in Section 2. Section 3 introduces basic concepts of the Ethereum protocol, a blockchain specification which we will use for our implementation. Sturm et al. proposed in [24] [23] a concept for a reusable blockchain-ready software artefact which allows to interpret simplistic process models and thus drive the interorganizational process autonomously. We build upon this work and describe the preliminaries in Section 4. However, for trusted and advanced data-based routing, we have to extend this architecture in Section 5, so that data and in-process variables are also kept in a decentralized fashion. Section 7 summarizes related work before Section 8 concludes the paper.

2 Novel Interoperability: Decentralized Control

In this section, we propose a novel form of interoperability to execute interorganizational workflows. We will briefly introduce the present forms and point to weaknesses in regard of requirements to interorganizational process execution. The novel form is contextualized in terms of modelling and implementation aspects. In the matter of modelling, we refer to *BPMN (Business Process Modelling and Notation)* [21] as the de-facto standard in research due to the frequent usage.

Conceptional Architectures and Interoperability Van der Aalst identified six different forms of interoperability between workflows to establish an interorganizational collaboration [26]. Each of them comes with certain drawbacks which we will discuss in this section. *Subcontracting* (i) assumes a hierarchical structure of participants, *chained execution* (ii) requires the process to consist of sequentially ordered parts and the *extended-/case transfer* (iii/iv) restricts full parallelism, as cases remain at one business partner solely at the same time [26].

Capacity sharing (CS) and *loosely coupled workflows (LCW)* as two remaining forms promise higher flexibility and are in the spotlight now. Instead of portioning the workflow, CS assumes centralized control (cf. Figure 1, left). One central workflow engine distributes tasks directly to resources of business partners, based on one global workflow. With the use of LCW (cf. Figure 1, right), each participant have a local workflow running which is connected to related workflows of other participants. They synchronize at certain points to ensure the correct execution of the overall business process.

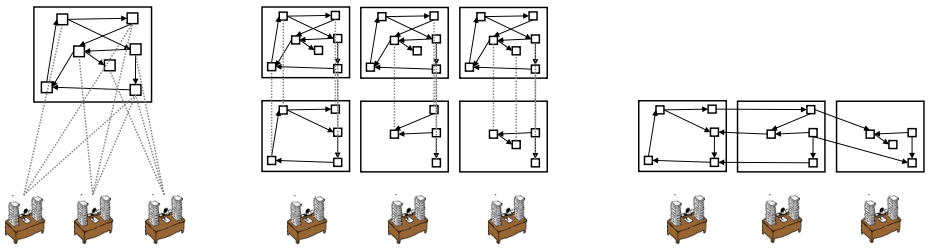

Fig. 1. Different forms of interoperability: Capacity sharing (left), Loosely coupled workflows (right) and Decentralized control (middle). Illustration based on [26].

We identify basic requirements to interorganizational process management in literature (cf. Table 1) and point out the downsides of CS and LCW to stress the necessity for a novel approach. A *common IT infrastructure* refers to a data storage or processing environment, which is accessible within a well-defined network, but secured against the global internet, e.g. a company-wide network drive which is protected behind a firewall. A centralized WFMS holds processes and data centralized by definition (CS), whereas LCWs cannot satisfy this requirement due to the geographical and logical separation of data. The preservation of

Table 1. Basic Requirements for Interorganizational Process Execution

Criteria	CS	LCW	Decentralized control
Common IT Infrastructure	+	-	+
Autonomy	-	+	+
Global Monitoring	+	-	+
Privacy	-	+	+
Trust	-	-	+

the *autonomy* demands that participants must be able to apply steady changes to internal workflows at any time [12]. LCW as implemented in [13] supports autonomy where local process model fragments of participants are connected with an event system, whereas CS uses an agreed-upon global workflow in a centralized WFMS where ad-hoc changes of single participants are hard to apply. Global *monitoring* is essential to query the progress of a particular workflow instance. Trivially, monitoring is easy within a centralized WFMS (CS), but with LCWs, increasing cardinality of participants raises both importance and complexity of process monitoring at the same time [14]. The third requirement refers to *privacy* [14]. Companies want to protect business secrets for reasons of preserving competitive edges or are obliged to keep data private because of legal constraints (e.g. GDPR). Using LCWs, only the tasks which are responsible for message exchange are necessary to be publicy visible. Therefore, LCW surpasses CS, where all tasks remain public in the global workflow. Trust was detected recently as impeding factor for establishing interorganizational workflows. That is, when a service provider is not trusted to host the infrastructure (CS) or a collaborating party corrupt historic execution logs to their advantage [27]. The latter concerns LCWs, because participants have full control over their fragment of the overall workflow.

We propose a novel form of interoperability, which solves the issues of global monitoring without centralizing the responsibility and which preserves local autonomy and the privacy of participants: In *decentralized control* (cf. Figure 1, middle), a trusted, decentralized and fault tolerant network builds the common IT infrastructure. Each participant holds the process data (models, instances/cases, in-process variables) locally. At first, this introduces redundant copies of data which may not always correspond to each other due to intentional manipulation or technical faults and hence is untrustworthy (byzantine faulty). Consensus mechanisms and especially byzantine fault tolerant algorithms can deal with a certain number of malfunctioned or wilful attacking nodes and ensure that honest and reliable nodes will automatically agree on a common state [3]. This adds a trustworthy layer of global monitoring and still avoids centralization. In the local environment, the participants are free to decouple the local confidential workflow from the global process so that autonomy and privacy is preserved. In the remainder of this paper, we implement a decentralized system leveraging on blockchain technology and the Ethereum protocol in particular.

Section 3 discusses in detail, how the components of Ethereum address the stated requirements.

Modelling Different paradigms for modelling interorganizational processes were proposed. Message Sequence Charts are used [2], when the sole interaction of workflows is of particular interest. Workflow nets are used to model the global workflow in the Public-to-Private approach [1]. This top-down approach is still current in state of the art methodologies, e.g. when high level milestones are defined first, before interaction of business partners with these milestones are identified. Then, global behavioural interfaces of participants are detected before they get finally linked to local executable workflows in the LCWs style [28].

The approaches correspond in a contract-like definition of the intermediate collaboration interface which must be respected by local workflows as described with LCWs. BPMN reflects this conceptual architecture in its various diagrams: Local workflows are modelled as *private executable processes* ([21] §10.2.1.1) which are interpreted in the WFMSs. For communication issues, *public processes* ([21] §10.2.1.2) define the behavioural interface, i.e. the global visible tasks of one single participant which are connected to global visible tasks of another participant (cf. Figure 3). The overall interorganizational workflow, is shown in a *choreography* ([21] §11) which is conceptually located between the participants. Choreographies are not directly executed by a central controller, but rather specify the to-be interactions of local workflows and focus on the required exchange of messages.

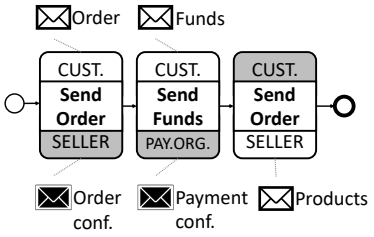

Fig. 2. BPMN Choreographie [28]

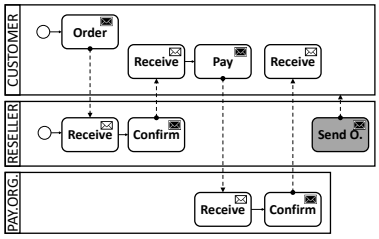

Fig. 3. *Send order* is not executed, because the *Seller* is not informed

Weske et al. describes certain issues that come with choreographies [28], which are a consequence of the tight coupling with LCWs. For instance, the choreography in Figure 2 is not *enforceable* ([21] §11.5.6), because the initiator (*Seller*) of the message exchange *Send order* is not involved in the previous *Send Funds* and therefore not informed that the payment was made. Enforceable choreographies must contain such message exchanges explicitly and unnecessarily blow up the models. This issue is due to the message-centric modelling in choreographies and the lack of a centralized controlling instance.

Decentralized control promotes to model and execute interorganizational executable processes which avoids these message-centric issues by nature. All participants get informed on state changes of the process automatically as the process progress and in-process variables are globally available, hence there is no need to focus on message exchanges. We will therefore concentrate on the interorganizational control-flow and model the intermediate contract as private executable process in Section 4 and Section 5 . In essence, no BPMN diagram is specialized on the control-flow between cross-enterprise organizational units, e.g. participants are usually modelled within BPMN pools, but the control-flow cannot leave the pool's boundaries ([21] §9.3). This and the different diagrams show that BPMN is significantly influenced by the LCW interoperability and consequently cannot directly support the novel interoperability. However, we will still use BPMN elements and take the officially specified semantics, whenever possible. At some point, we will have to reinterpret elements in the proper meaning of BPMN.

Implementation Aspects of Interorganizational Workflows Interorganizational workflows and LCWs in particular can be implemented using web services (cf. WS-BPEL, the Business Process Execution Language). This realises an integration of two separate machines of different companies to automatically exchange data, but not process enactment in the sense of decoupling application programs from the process logic to achieve the flexibility of work item distribution or global data-based routing of the process flow. That is rather achieved by interpreting process models in a process-aware management tool.

In decentralized control, we implement an executable process model in a decentralized WFMS. Speaking of implementing processes, Dumas et al. separates between human tasks and service tasks [5]. The former are rather manual tasks which are hard to track automatically, e.g. phone calls or manual exchange of raw materials in production lines. The latter is for instance a small program or script which sends an email or updates in-process data values. Hence, a WFMS serves as distributor of work items to the respective resource on the one hand, but may also be responsible for process automation. In this paper, we focus on human tasks and ensuring proper execution with data-based routing, but do not automate certain tasks. However, it has been already proven, that Ethereum can also accomplish basic automated service tasks [16], but the solution requires to write program code during process modelling.

3 Selecting Blockchain as Decentralized Environment

Blockchains are a novel possibility to build large-scale byzantine fault tolerant networks. Ethereum can execute decentralized applications on top and has proven to be able to run business processes [16][27]. Hence, it seems to be a suitable solution for implementing decentralized control interoperability. We introduce basic concepts of Ethereum now, but concentrate on the essential details to keep the blockchain as black-boxed as possible for readability purposes.

On a very high-level view, we can describe the blockchain using four perspectives. From a *technical perspective*, a blockchain is a protocol running within an open peer-to-peer network everyone can join by running a client software. The protocol includes a gossip functionality so that connected participants and thus at some time the whole network is notified about necessary information. From a *functional perspective* the blockchain is a distributed append-only data storage, where a consensus mechanism allows the network to decide over new data (new blocks) autonomously. That includes, to select the responsible entity for proposing a new block and to decide on the correctness which is determined by a protocol-given ruleset. As an example, the ruleset in the Bitcoin protocol includes that a transaction is propagated only if the senders' balance is beyond the amount of monetary units he wants to transfer. The winner of a laborious puzzle is the chosen one to propose a new block (Proof-of-work) which must comply with the ruleset so that the block is appended to the chain. The *integrity perspective* of the blockchain practically prevents that historical data gets modified or corrupted ex post. Blocks include a hash value of the previous block so that a chain towards the initial (genesis) block is established. Data modifications are easy to detect, because the hash value of the respective block is updated and propagated through the chain towards the latest block. Convincing the network that the corrupted sibling chain is the valid one is prevented due to the laborious calculation of blocks and the rule to follow the longest chain in the network. We will also exploit the *extensibility perspective* of blockchains. For our purposes, we define this as the possibility to modify the ruleset for database inserts. The level of extensibility culminates in the Ethereum blockchain where arbitrary (turing-complete) programs (*smart contracts*) can be executed.

Speaking in BPM terms, the technical perspective provides a common IT infrastructure. The blockchain can be installed as a private chain which allows the required access control to the processes and data. The functional perspective prevents centralization, because the consensus mechanism ensures that the participants decide as group over new data, i.e. the process state, autonomously. The integrity perspective establishes a trusted layer. Each participant stores a copy of the data locally and is free to apply manipulations to his favour, but after the data was propagated and accepted by the network such manipulations are detected easily. We rely on the extensibility perspective and design a smart contract to define a process (model)-aware ruleset and so enforce model compliance including automated data-based routing during the execution. In essence, the blockchain can serve the requirements of interorganizational process management which were identified in Section 2.

The operability of a blockchain is not safeguarded by design but ensured during run time. Attacks are theoretically possible, but with consensus mechanisms which are driven with an economical background, attempted manipulation causes the loss of money whereas canonical behavior is awarded. This is funded by fee-based transactions and on-chain storage. To minimize the costs for process execution, we focus on a small set of BPMN elements instead of supporting a full-blown modelling language.

4 On-Chain Interpreted Execution of BPMN-Processes

The next two sections introduce the design of our decentralized control implementation. Semantics for certain BPMN modelling elements are defined to model interorganizational processes which are then executed within a decentralized management system. In [24], a novel approach is presented to interpret BPMN-notated models on the Ethereum blockchain. We will briefly recap the preliminaries and extend this architecture in Section 5 for our purposes with data values to facilitate data-based routing of the control-flow.

4.1 Execution Semantics

The BPMN specification defines the execution semantics of the modelling language in textual form with references to the token game in a petri net, but a formal definition regarding the execution is missing for some elements in the specification, especially in the interorganizational context. BPMN uses *pools* to model participants ([21] §7.3.1) but do not reference them in the section regarding the execution semantics ([21] §13). Further on, the meaning of *lanes* is not specified intentionally ([21] §10.8). In consequence, multiple interpretations can be found in different systems: Sometimes, pools are used for a short process description and process participants are depicted in lanes [16] or Camunda[4] ignores pools and lanes completely and uses an own account system.

To avoid any discrepancies, we propose the execution semantics for rudimentary BPMN elements and adopt them to the needs for our architecture. The upcoming definitions regarding the execution semantics are built upon *requirements* or *pre-/post-conditions* to model the intermediary contract of the interorganizational process. Informally spoken, previous to the execution of a task, we check that all preconditions of this task are fulfilled, for instance the preceding task was executed, the task is not locked due to the execution of concurrent tasks or all data conditions are fulfilled (cf. Section 5). As stated, we do not claim to be BPMN compliant. For instance, we use pools as the process container and lanes are mapped to the responsible participants so that the control-flow can be distributed over all participants.

4.2 Sequence and Parallel Gateway

We define an interorganizational process IOP as a 5-tuple

$$IOP = (\mathcal{T}, \mathcal{G}, \Sigma, \mathcal{F}, \mathcal{L}),$$

referring to tasks, gateways, sequence flows, fulfilled tasks and locked tasks.

Tasks and Gateways. $\mathcal{T} = \tilde{\mathcal{T}} \cup \{\epsilon\}$ unions the set of the tasks $\tilde{\mathcal{T}}$ in the model with the start node denoted as ϵ, and $\mathcal{G}$ comprises the set of gateways. The function $\gamma : \mathcal{G} \to \{\times, \circ, +\}$ maps gateways to their respective type in the model: $+$ for a parallel gateway, $\times$ for an exclusive gateway and $\circ$ for an inclusive gateway (cf. Section 5).

[4] A widely used open-source BPMS: https://camunda.com/

Control-Flow. The set

$$\Sigma = \{\sigma_i | i \in \mathbb{N}\} = \{(\sigma_i^1, \sigma_i^2) \in \mathcal{T} \cup \mathcal{G} \times \mathcal{T} \cup \mathcal{G} \,\big|\, \Xi, i \in \mathbb{N}\}$$

with condition Ξ describes the control-flow. Ξ claims for each tuple $(\sigma^1, \sigma^2) \in \Sigma$ a corresponding sequence flow from σ^1 to σ^2 in the given BPMN model.
We define a helper function $\kappa : \mathcal{G} \to \{\prec, \succ\}$ for determining if the gateway is a forking gateway ($\prec$) or a merging gateway ($\succ$) respectively:

$$\kappa : g \mapsto \begin{cases} \prec, & |\{(g,t) \in \Sigma \;\big|\; t \in \mathcal{T}\}| > 1 \\ \succ, & |\{(g,t) \in \Sigma \;\big|\; t \in \mathcal{T}\}| = 1 \end{cases}$$

For readability purposes, notation $g_{\prec}$ is used for forking gateways ($\kappa(g) = \prec$) and ${}_{\succ}g$ describes merging gateways ($\kappa(g) = \succ$). The gateway type is denoted in superscript, e.g. a merging exclusive gateway would be denoted as ${}_{\succ}g^{\times}$ ($\kappa(g) = \succ$, $\gamma(g) = \times$).

Preconditions. For a task t to be executable, the following conditions must apply. The task is not locked ($t \notin \mathcal{L}$, cf. data-based postconditions in Section 5) or finished ($t \notin \mathcal{F}$, cf. postconditions below). Henceforth, let Θ be an abbreviation for $t \notin \mathcal{F} \cup \mathcal{L}$. Further on, a task execution is bound to requirements, i.e. tasks which must have been potentially executed before. We define requirements as function $\rho : \mathcal{T} \to \mathfrak{P}(\mathcal{T})$[5]. In the following definition for ρ, let $\tilde{t} \in \mathcal{T}$ and $g \in \mathcal{G}$.

$$\rho : t \mapsto \begin{cases} \emptyset, & (\epsilon, t) \in \Sigma \\ \{\tilde{t}\}, & \exists \tilde{t} : (\tilde{t}, t) \in \Sigma \\ \{\tilde{t}\}, & \exists \tilde{t}\ \exists g_{\prec} : \big((\tilde{t}, g_{\prec}) \in \Sigma\big) \wedge \big((g_{\prec}, t) \in \Sigma\big) \\ \tilde{\mathcal{T}}, & \exists {}_{\succ}g^{+} \in \mathcal{G} : ({}_{\succ}g^{+}, t) \in \Sigma \end{cases}$$

with $\tilde{\mathcal{T}} = \{\tilde{t} \,|\, \{(\tilde{t}, {}_{\succ}\, g^{+}), ({}_{\succ}g^{+}, t)\} \subset \Sigma\}$.

Consider Figure 4. Task A does not include any requirements, because it directly follows the start node (cf. line 1 of ρ). Task A itself is requirement for B, because a sequence flow exists in between (line 2). Before executing the tasks after the forking gateway (C and D), B must have been executed (line 3). And finally, the task after the merging gateway (G) must wait for the final tasks of the respective flows, which are F and D in the process model in Figure 4 (line 4).

We define the preconditions for an execution of t as function $P : \mathcal{T} \to \{\checkmark, \times\}$ now. Again, let $\tilde{t} \in \mathcal{T}$ and $g \in \mathcal{G}$ unless otherwise stated. $P(t) = \checkmark$ indicates that the task t is executable and vice versa $P(t) = \times$ indicates that the task t is not executable.

[5] $\mathfrak{P}(\mathcal{T})$ describes the cartesian product of $\mathcal{T}$

$$P : t \mapsto \begin{cases} \checkmark, & \exists \tilde{t} : (\tilde{t}, t) \in \Sigma \wedge (\rho(t) \subseteq \mathcal{F}) \wedge \Theta \\ \checkmark, & \exists g^{+}_{\prec} : (g^{+}_{\prec}, t) \in \Sigma \wedge (\rho(t) \subseteq \mathcal{F}) \wedge \Theta \\ \checkmark, & \exists_{\succ} g^{+} : (_{\succ} g^{+}, t) \in \Sigma \wedge (\rho(t) \subseteq \mathcal{F}) \wedge \Theta \\ \text{✗}, & t \in \mathcal{F} \vee t \in \mathcal{L} \\ \text{✗}, & else \end{cases}$$

Organizational Perspective. We use BPMN lanes to determine the respective resource, i.e. the actor responsible for executing a task. The interaction with blockchain technology is based on cryptographic public keys which are used for the lane's name. In Figure 4, 0x01E... is responsible for the tasks B and D. We omit the mathematical foundations here due to limited space, but the system ensures that tasks are only executable by the respective resource.

Postconditions. After a task t was executed, denoted as $t_{\checkmark}$, it is not longer executable and included in the set of finished tasks:

$$\mathcal{F} = \mathcal{F} \cup \{t_{\checkmark}\}$$

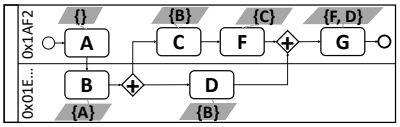

Fig. 4. A process with requirements in parallelograms.

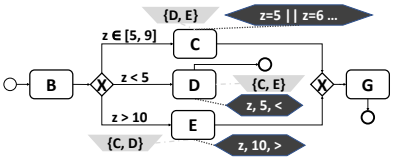

Fig. 5. A process with decisions in dark hexagons and competitors in light trapezoids.

5 Modelling and Executing Data-based Decisions

Data is essential to processes from an application-oriented point of view and one of five important aspects which were identified and conflated in the multi-perspective process model [10]. In the context of blockchain-based process execution, the functional and behavioural perspective were focused first and also some organizational aspects are rudimentary considered. In this section we focus on the the informational perspective by emerging data-based decisions to control the flow during the execution of a process. In contrast to current systems, where global data is centralized (CS) or not available (LCWs), the blockchain as operator is responsible here for holding all process data. The blockchain must consequently be the authority which decides on the control-flow in case of data-based decisions which are automatically triggered or based on human input. We

refer to the definitions given in Section 4 and extend these in terms of data-based decisions.

5.1 Modeling decisions in BPMN

The operation of a business process is rarely predictable a priori but often determined dynamically based on internal data or external interactions and decisions. BPMN provides constructs to model the split of the control-flow based on conditions (data-based routing). Here, we focus on *exclusive gateways* ([21] §10.6.2), *inclusive gateways* (§10.6.3) and *parallel gateways* (§10.6.4).

To illustrate the usage of data-based routing, we refer to Figure 5. After the execution of task B, the current value of z determines the upcoming valid execution path(s). The conditions are labeled on the gateway's outgoing sequence flows, as example C is enabled for $z = 5$, D is enabled for $z = 0$ or E is enabled for $z = 15$.

5.2 Formalization of the Execution Semantics

Data storage. We extend the interorganizational process IOP by a global data store Φ, with

$$\Phi = \{\phi_i | i \in \mathbb{N}\} = \{(\phi_i^1, \phi_i^2) | i \in \mathbb{N}\}$$

Φ is responsible for holding all in-process data variables which may be consulted in data-based decisions during process execution, where ϕ^1 refers to the variable (e.g. z in Figure 5) and ϕ^2 is the current value which may be updated on occasion. Note that we refer to the value of ϕ_i only at time of the evaluation and neglect changes over time.

Conditional Control-flow. If a gateway splits the control-flow in BPMN, the labels on the outgoing sequence flows determine the set of valid execution paths. For this purpose, we introduce the set of *decisions* $\mathcal{D}$, with

$$\mathcal{D} = \{d_i | i \in \mathbb{N}\} = \{(\phi, \lambda, o)_i | i \in \mathbb{N}\},$$

where $\phi \in \Phi$ is the global payload and λ is the reference value on the sequence flow. We define λ here informally as number, text or numerical interval. λ is evaluated against of the value of the global payload ϕ^2 using a suitable relational operator o (e.g. $>, <, =$ or $\in$ if λ is an interval). The function $\mu : \Sigma \to \mathcal{D}$ maps sequence flows originating from an inclusive ($g^{\circ}_{\prec}$) or exclusive ($g^{\times}_{\prec}$) forking gateway to a decision. Consider Figure 5 as an example. Let σ_2 be the upper sequence flow from B to C.

$$\Phi = \{\phi_0\} = \{(z, 15)\}$$
$$\mathcal{D} = \{(\phi_0, [5, 9], \in), (\phi_0, 5, <), (\phi_0, 10, >)\}$$
$$\mu(\sigma_2) = d_0 = (\phi_0, [5, 9], \in)$$

New Preconditions. For a task t that follows a forking inclusive or exclusive gateway ($g^{\circ,\times}_{\prec}$), the assigned condition $\mu((g^{\circ,\times}_{\prec}, t))$ must be fulfilled, denoted as $\mu_{\checkmark}((g^{\circ,\times}_{\prec}, t))$. We extend the definition of $P : \mathcal{T} \to \{\checkmark, \times\}$ towards the data-based decision-aware preconditions. Unaffected cases from the definition of P in Section 4 remain unchanged.

$$P : t \mapsto \begin{cases} \checkmark, & \exists g^{\circ,\times}_{\prec} : (g^{\circ,\times}_{\prec}, t) \in \Sigma \wedge (\rho(t) \subseteq \mathcal{F}) \\ & \wedge \mu_{\checkmark}((g^{\circ,\times}_{\prec}, t)) \wedge \Theta \\ \checkmark, & \exists_{\succ} g^{\circ,\times} : (_{\succ}g^{\circ,\times}, t) \in \Sigma \wedge (\rho(t) \subseteq \mathcal{F}) \wedge \Theta \end{cases}$$

New Postconditions. Having executed task $t_{\checkmark}$, the postconditions are now threefold: (i) finishing: uniting $\{t_{\checkmark}\}$ with $\mathcal{F}$ (cf. Section 4), (ii) locking: uniting of concurrent tasks with $\mathcal{L}$, (iii) preparatory unlocking: uniting the final tasks of enabled paths with the requirements of the task following the corresponding merging gateway.

Consider (ii) locking. If the executed task $t_{\checkmark}$ succeeds a forking exclusive gateway ($(g^{\times}_{\prec}, t_{\checkmark}) \in \Sigma$), other succeeding tasks of this gateway $\tilde{\mathcal{T}} = \{\tilde{t} \in \mathcal{T} | (g^{\times}_{\prec}, \tilde{t}) \in \Sigma\} \setminus \{t_{\checkmark}\}$ get locked, because BPMN allows only one execution path after an exclusive gateway:

$$\mathcal{L} = \mathcal{L} \cup \tilde{\mathcal{T}}$$

Additionally (cf. (iii) preparatory unlocking), in case of $t_{\checkmark}$ following $g^{\circ,\times}_{\prec}$, i.e. a forking inclusive or exclusive gateway, the corresponding merging gateway must be aware of all valid conditions, to synchronize on all enabled execution paths. We therefore add the final task of the respective enabled path to the requirements of the task following the merging gateway ($\tilde{t}$). Let ${}_{\prec^{\circ,\times}}T_{\checkmark}$ denote the executed tasks following a forking inclusive or exclusive gateway:

$${}_{\prec^{\circ,\times}}T_{\checkmark} = \{t_{\checkmark} | \exists g^{\circ,\times}_{\prec} \in \mathcal{G} \wedge (g^{\circ,\times}_{\prec}, t_{\checkmark}) \in \Sigma\}$$

For all tasks in ${}_{\prec^{\circ,\times}}T_{\checkmark}$, there exists a path to the corresponding merging gateway[6], i.e. $\exists k : \{(t_{\checkmark}, \sigma^2_0), (\sigma^1_1, \sigma^2_1), ..., (\sigma^1_{k-1}, \sigma^2_{k-1}), (\sigma^1_k, {}_{\succ} g^{\circ,\times}), ({}_{\succ}g^{\circ,\times}, \tilde{t})\} \in \Sigma$, where $\sigma^2_i = \sigma^1_{i+1} \forall i < k$ and $i, k \in \mathbb{N}$. We add all tasks before the merging gateway to the requirements of task $\tilde{t}$:

$$\rho(\tilde{t}) = \rho(\tilde{t}) \cup \sigma^1_k$$

6 Implementation

The implementation of decentralized control is subdivided into the implementation of the decentralized system and the local workflow management. As proof-of-concept, we implemented a decentralized system as web application based on Ethereum. The code and a screencast is available on Github[7]. Instead of calling

[6] for block-structured processes

[7] https://github.com/sensati0n/Ethereum-4-Decentralized-Control

the smart contract functions from the demo application, they can be called programmatically and integrated in suitable WFMSs like Camunda. In this section, we briefly present the usage and refer to the respective module in brackets.

At first, each collaborator is registered in the application (`add_collaborator`). After the interorganizational process model was uploaded as *.bpmn-file, it gets parsed and the tasks with its requirements are written onto the smart contract (`xml_parser`). Thereby, the account in the respective lane is assigned to the tasks. Additionally, all data values on gateway's outgoing sequence flows describing a conditional control-flow are parsed and written as global payload into the smart contract. A user can complete his tasks (`complete_task`), when his account is selected in `Select Process Collaborators` (and all conditions are fulfilled). Values of global payloads can be changed in `change_global_payload`.

7 Related Work

Decentralized control as new form of interoperability promotes locally distributed data retention with a decentralized consensus finding towards a logically centralized single source of truth. In our paper, the implementation relies on Ethereum blockchain as decentralized network which acts as the source of truth and cares for process execution. We build upon [24] and in this section we glance at related work on blockchain-based process execution.

Approaches can be categorized by software engineering concerns. For instance, [27] compiles BPMN choreographies into a blockchain-readable program code fragment on the one hand, whereas [23][24][17] forged ahead with providing a process-aware environment on the blockchain a priori and interpret the process model thereon. Besides, interpreting process models, [25] follows a model-driven engineering code-generation approach. [6] abstracts from any particular blockchain protocol and uses adapters to monitor process execution steps on-chain. We conclude with the categorization according to BPM concerns. Firstly, different modelling paradigms were addressed, e.g. declarative processes [18], BPMN choreographies [11], artefact-based approaches [8], rule-based solutions [9] or the use of state charts [20]. A BPMN extension is developed by [6]. Also different perspectives of process models are addressed. For instance, concerning the organizational perspective, [15] provides a flexible approach to bind actors to tasks at runtime whereas [23] shows how to integrate such resources statically. Besides profiting from actively driven execution, the immutable ledger is also used as storage during monitoring as proposed in [4] [22]. Not only generic solutions, but also domain specific studies were published, e.g. [7] has proven the feasibility of integrating blockchain technology in a cross-organizational process in the financial sector. Some of the mentioned publications were reassembled into Caterpillar [16], a sophisticated blockchain-based WFMS which supports many BPMN concepts and service tasks on the one hand, but process modellers are concerned with writing smart contract code to support data-based decisions on the other hand.

8 Conclusion

With decentralized control, we have introduced a new form of workflow interoperability in interorganizational process execution. It is based upon a decentralized network, which is secured with byzantine fault tolerant algorithms. We used BPMN to model the workflows, albeit the notation does not support the form of interoperability directly. We avoid this issue by defining own semantics to basic BPMN constructs. To support data-based routing, we have extended the proposal of [24]. Instead of implementing bilateral machine-to-machine message-exchanges using web services, we rather record state changes of the process on the blockchain, so that every participant can be informed easily. We have shown that our approach outperforms present forms of interoperability w.r.t. basic requirements to interorganizational process execution in literature.

The current solution to decentralized control is affected from certain drawbacks. As stated, BPMN is currently not a tailored solution, because modelling global and enforceable interorganizational processes is mainly message-driven in BPMN. Research has to evaluate, if the novel form of interoperability requires novel forms of interorganizational process modelling. The Ethereum blockchain has proven to facilitate decentralized control, but might not be the most suitable solution. As an example, consensus finding is a requirement, but the included consensus mechanism in Ethereum (Proof-of-work) was designed for large-scale consensus finding in global networks and ensures stability due to a monetary incentive mechanism. Both artefacts do not directly support BPM and even may impede the implementation: collaborations do not necessary consist of a high amount of participants and participants should not be confronted with paying for process execution. We will follow a top-down approach in future work and define requirements to decentralized control, before suitable algorithms and strategies (e.g. small-scaled byzantine fault tolerant consensus mechanisms) are evaluated thoroughly, instead of using a generalized out-of-the-box solution like Ethereum.

References

1. van der Aalst, W., Weske, M.: The p2p approach to interorganizational workflows. In: Proc. of the 13th International Conference on Advanced Information Systems Engineering. CAiSE '01, Berlin, Heidelberg (2001)
2. van der Aalst, W.: Interorganizational workflows: An approach based on message sequence charts and petri nets. Systems Analysis Modelling Simulation (1999)
3. Castro, M., Liskov, B.: Practical byzantine fault tolerance. In: Proc. of the Third Symposium on Operating Systems Design and Implementation. OSDI '99 (1999)
4. Di Ciccio, C., Cecconi, A., Mendling, J., Felix, D., Haas, D., Lilek, D., Riel, F., Rumpl, A., Uhlig, P.: Blockchain-Based Traceability of Inter-organisational Business Processes (2018)
5. Dumas, M., Rosa, M.L., Mendling, J., Reijers, H.A.: Fundamentals of Business Process Management, Second Edition. Springer (2018)
6. Falazi, G., Hahn, M., Breitenbücher, U., Leymann, F.: Modeling and execution of blockchain-aware business processes. SICS Journal **34** (2019)

7. Fridgen, G., Radszuwill, S., Urbach, N., Utz, L.: Cross-organizational workflow management using blockchain technology (2018)
8. Hull, R., Batra, V., Chee, Y.M., Deutsch, A., Heath, F., Vianu, V.: Towards a shared ledger business collaboration language based on data-aware processes (2016)
9. Idelberger, F., Governatori, G., Riveret, R., Sartor, G.: Evaluation of logic-based smart contracts for blockchain systems. In: Rule Technologies. Research, Tools, and Applications (2016)
10. Jablonski, S., Bussler, C.: Workflow Management: Modeling Concepts, Architecture, and Implementation (1996)
11. Ladleif, J., Weske, M., Weber, I.: Modeling and enforcing blockchain-based choreographies. In: BPM - 17th International Conference, BPM 2019
12. Legner, C., Wende, K.: The challenges of inter-organizational business process design - A research agenda. In: Proc. of the 15. European Conference on Information Systems, ECIS 2007
13. Lindert, F., Deiters, W.: Modelling inter-organizational processes with process model fragments. CEUR Workshop Proceedings, vol. 24 (1999)
14. Liu, C., Li, Q., Zhao, X.: Challenges and opportunities in collaborative business process management: Overview of recent advances. Inf. Syst. Frontiers **11** (2009)
15. López-Pintado, O., Dumas, M., García-Bañuelos, L., Weber, I.: Dynamic role binding in blockchain-based collaborative business processes. In: CAiSE 2019
16. López-Pintado, O., García-Bañuelos, L., Dumas, M., Weber, I., Ponomarev, A.: Caterpillar: A business process execution engine (...). Softw. Pract. Exp. **49** (2019)
17. López-Pintado, O., Dumas, M., García-Bañuelos, L., Weber, I.: Interpreted execution of business process models on blockchain (2019)
18. Madsen, M.F., Gaub, M., Høgnason, T., Kirkbro, M.E., Slaats, T., Debois, S.: Collaboration among adversaries: Distributed workflow execution on a blockchain. Symposium on Foundations and Applications of Blockchain (2018)
19. Merz, M., Liberman, B., Muller-Jones, K., Lamersdorf, W.: Interorganisational workflow management with mobile agents in cosm (1996)
20. Nakamura, H., Miyamoto, K., Kudo, M.: Inter-organizational business processes managed by blockchain. In: Web Information Systems Engineering – WISE 2018
21. OMG: Business Process Model and Notation (BPMN), Version 2.0.2 (2013)
22. Prybila, C., Schulte, S., Hochreiner, C., Weber, I.: Runtime verification for business processes utilizing the bitcoin blockchain. CoRR (2017)
23. Sturm, C., Scalanczi, J., Schönig, S., Jablonski, S.: A blockchain-based and resource-aware process execution engine. FGCS Journal **100** (2019)
24. Sturm, C., Szalanczi, J., Schönig, S., Jablonski, S.: A lean architecture for blockchain based decentralized process execution. In: Business Process Management Workshops. Springer International Publishing (2019)
25. Tran, A.B., Lu, Q., Weber, I.: Lorikeet: A model-driven engineering tool for blockchain-based business process execution (...). In: BPM (2018)
26. van der Aalst, Wil: Process-oriented architectures for electronic commerce and interorganizational workflow. Information Systems **24** (1999)
27. Weber, I., Xu, X., Riveret, R., Governatori, G., Ponomarev, A., Mendling, J.: Untrusted business process monitoring and execution using blockchain. In: Business Process Management - 14th International Conference, Proceedings (2016)
28. Weske, M.: Business Process Management - Concepts, Languages, Architectures, Third Edition. Springer (2019)