DECENTRALIZED BUSINESS PROCESS MODELING AND INSTANCE TRACKING SECURED BY A BLOCKCHAIN

Research paper

Härer, Felix, University of Bamberg, Bamberg, Germany, felix.haerer@uni-bamberg.de

Abstract

For supporting the conceptualization and the management of enterprise models in a decentralized manner, this paper introduces an approach based on model versioning and blockchain technologies. The main contribution is twofold, consisting of a., the creation of models for inter-organizational business processes in a decentralized environment, and b., means for tracking process instances using meta-data at run time. Models for business processes, workflows, and instance states are collaboratively created as part of a decentralized architecture. Based on this approach, a hierarchical versioning and modeling approach is employed in order to create and manage public and private models in a transactional fashion. For forming relationships among decentralized participants, semi-formal models linked to a blockchain are suggested. The approach is evaluated with a supply chain use case and demonstrated in an implemented modeling tool.

Keywords: Collaborative Modeling, Decentralization, Model Management, Blockchain, Model Versioning, Modeling Tools.

1 Introduction

Interactions across organizations for the purposes of collaboration are one of the drivers of today's large-scale value networks. For fostering value creation in inter-organizational collaborations, participants develop a shared understanding of business processes and workflows. If interactions can be sufficiently documented in process definitions, conceptual modeling (Karagiannis et al. 2016) is often used for handling the complexity of the represented interactions through models, e.g. in BPMN collaboration diagrams (OMG 2014). Methodologies and software modeling tools are traditionally based on centralized architectures (Maróti et al. 2014) and have advanced towards web architectures (Nicolaescu et al. 2017). With recent decentralization trends (Ferdinand et al. 2016, Brenig et al. 2016, Nærland et al. 2017), however, individual businesses might act as *peers* and participate in networks without central coordination. In a decentralized environment, achieving consensus on a shared understanding of processes is a major challenge. Processes are planned from the local perspective of each individual participant and are at the same time required to fulfill a role in a global choreography. Participants rely on information exchanged among each other, as they have no central coordinator, and require it to be consistent and dependable. As an example, in an agile procurement process in an Industry 4.0 context, new tier 2 and 3 vendors may be added at all times (Nicoletti 2017). By means of decentralized planning of the purchase order process, vendors without long-established relationships have the ability to join the process instantly, based on information from process models secured by a blockchain. Participants can reliably verify the integrity of process models on their own, allowing for collaborations to be built based on models, reducing coordination cost with the potential of making the processes part of open value networks.

Conceptual modeling can contribute in such a scenario through process modeling, however, models need to be managed in a way which allows for reaching a consensus among decentralized participants, regarding the process and its instantiation. Existing approaches for managing models are not concerned with forming relationships and agreements, since they are often rooted in a centralized paradigm, leaving aside the potential for inter-organizational agreements in a decentralized environment.

Therefore, two research questions follow:

1. How can the creation of business process models among decentralized participants be managed providing that participants can collaboratively model and agree on models?

2. How can decentralized participants track instances of agreed-upon models?

By securing models with a blockchain, the use of conceptual modeling may become viable in areas where the integrity of a model is critical for security reasons. In cases where an external actor is presented with a model and is required to act on it without a "trusted third party", the integrity of the model can be verified. Models can be safeguarded against manipulation to prevent damages, e.g. models of procedures in response to critical infrastructure failures or models of manufacturing processes. In cases where a complex system creates or modifies models autonomously, models can be monitored for tracking changes or used for collaborations, e.g. with decentralized autonomous organizations.

Technologically, blockchains (Notheisen et al. 2017) have been suggested for supporting interactions between businesses through transactions, which have the notable property of being trust-free (Greiner and Wang 2015, Beck et al. 2016), i.e. integrity and immutability of any transaction is cryptographically secured, based on a decentralized peer-to-peer network. While implementations for agreement procedures exist through the concept of smart contracts (Szabo 1997) and the off-line storage of consistent enterprise models is solved by model versioning and versioning systems (Brosch et al. 2012), both aspects need to be combined in order to support interactions where participants are decentralized.

When implementing business transactions with smart contracts, the participants have to write formalized functions, most commonly using the procedural programming language Solidity (Buterin et al. 2017), for defining every parameter and all steps of possible interactions, which, for complex scenarios, is inconceivable at build time. Even for simpler cases, procedurally written contracts often contain bugs due to being overly specific and complex (Fröwis and Böhme 2017). Less-formalized abstractions might be useful for sharing information, negotiations, and resulting agreements. This paper suggests the use of semi-formal models, linked to a blockchain through smart contracts, instead of using smart contracts as procedural implementations of business transactions. The use of semi-formal models allows for the representation of domain knowledge through concepts of the domain, which can be as precise as the domain requires.

The remainder of this paper is structured as follows. Section 2 introduces foundations and related work with regard to collaborative modeling, peer-to-peer systems, and blockchains. Section 3 discusses the main approach with its components for modeling, collaboration, and instance tracking using a hierarchical versioning approach linked to a blockchain. In Section 4, a use case of a collaborative business process evaluates the approach and demonstrates it in a software tool. The paper concludes with a summary and outlook in Section 5.

2 Foundations

Foundational concepts regarding Collaborative Modeling and technical concepts of Peer-to-Peer Systems and Blockchains are briefly introduced in the following sub-sections.

2.1 Collaborative Modeling

Methods and tools for the collaborative creation and execution of models, as well as the management of models, e.g. using model versioning, are central in research related to collaborative modeling. Inter-organizational workflows and collaborative processes are foundational concepts, especially relevant in conjunction with the collaborative creation of models. Inter-organizational workflows provide a formalized method for modeling state-based public workflows, shared between organizations, from which private workflows are derived (van der Aalst and Wekse 2001). Petri-Nets are used as a basis and allow for simulation. The broader term Collaborative Business Process also encompasses workflow models; however, it is not a specific method, but a generalized concept based on modeling a collaboration of interacting process partners, e.g. as choreography. Fdhila et al. (2015) identify the following aspects of collaborative business processes: 1) *model* with interacting partners, using private, public and choreography models, 2) *approaches*, top down from a global choreography or bottom up from local models, 3) *partner selection*, static with a-priori known partners or dynamic with selection and mapping of partners at run time, and 4) *properties* for collaborative business processes. Properties are the *consistency* of implementations and observable behavior (Decker and Weske 2007), the *compatibility* of structure and behavior in reference to *soundness* described by van der Aalst et al. (2011), and the *realizability* of the process model of each partner (Fdhila et al. 2015). While the satisfiability of these properties may be proven for formalized models, this approach focuses on modeling with semi-formal notations, common in enterprise modeling (Bork and Fill 2014).

Methods for the collaborative creation of models exist for a number of modeling languages as well as purpose-built languages for collaboration. UML-based approaches, e.g. Hofreiter and Huemer (2008) and Villarreal et al. (2010), often rely on extensions by UML profiles. Dollmann et al. (2011) suggest a tool-based approach for modeling on one abstraction level, e.g. with event-driven process chains. CPM by Ryu and Yücesan (2007) extends to the execution of models in the area of manufacturing, including local and global views. In Adaptive Case Management (ACM), an approach by Hewelt and Weske (2016) relies on cases with process fragments which are re-combined while the case is handled, however, without global (choreography) models, according to ACM. For the collaborative creation of models, modeling tools can offer multi-user support, e.g. in meta-modeling platforms like MetaEdit+, ADOxx (Fill and Karagiannis 2013), or in workflow-oriented tools and engines like Camunda BPM or jBPM (Geiger et al. 2017), which also cover execution. In addition, specific model versioning tools (Brosch et al. 2012) in the form of (distributed) version control systems (DVCS) are researched together with methods. DVCS are suitable for asynchronous off-line modeling, where a consistent state of any number of models is created as *version* by a commit operation of a modeler and retrieved by others using a check-out or fetch operation. In contrast, near-realtime modeling (Derntl et al. 2015) propagates model changes to participants almost instantly and allows for ad-hoc interactions, where participating modelers are synchronously online. Another difference between these approaches concerns the extent of consistency guarantees. Since in near-realtime modeling, operations can occur at any time on all models and all model elements, no consciously defined versions of n models exist, meaning that the consistent state extends to individual models, but not to multiple models, where inter-model references between model elements may break when models are changed. For undoing such a change, an undo-operation on a model level does not suffice, since the state of other models can change concurrently. Methods of near-realtime modeling, e.g. operational transformation and extensible data types (Nicolaescu et al. 2017), operate on one shared resource, i.e. a text or arbitrary data type instances, possibly model elements. In DVCS, an undo-operation reverts the global state, e.g. including all models, to a previous consistent state, however, versions have to be created explicitly by commit. Analogously to the development of complex software systems, where developers extensively use asynchronous approaches in version control systems with manual conflict resolution (Chacon and Straub 2014) instead of synchronous software development tools with (near-)realtime capabilities (Liu et al. 2006, Sun and Sosič 1999), the development of complex collaborative processes in multiple inter-linked models may profit from an asynchronous approach with explicit versioning provided by a DVCS.

2.2 Peer-to-Peer Systems

Peer-to-Peer systems are a long-established enabling technology for decentralization in the area of distributed systems. Approaches like Distributed Hash Tables (Steinmetz and Wehrle 2005, p. 79) implement decentralized and direct communication among network participants, or peers. The properties of technical peer-to-peer-systems are well researched (Steinmetz and Wehrle 2005, Kurose and Ross 2017, van Steen and Tanenbaum 2017). In a peer-to-peer system, peers are autonomous and communicate directly with each other. On a system level, peers can therefore be self-organizing. In contrast to Client-Server-based system architectures, a system participant does not have a prescribed role as server or client. Peers fulfill both roles when interacting with each other directly. At any point, a peer may connect or disconnect from the system, i.e. peers enter and exit at run time. On the level of individual peers, build time aspects are relevant before participating in the system, as well as run time aspects when joining the system. On the global system level, build time and run time aspects overlap for this reason. In this fashion, peer-to-peer systems realize decentralization, where “once a node has joined the system, it can use a fully decentralized scheme for collaboration” (van Steen and Tanenbaum 2017, p. 91). Related to the context of Information Systems, peers can represent business entities or business process participants on a network level, collaborating in an inter-organizational manner for realizing business transactions.

2.3 Blockchains

Blockchains have been proposed for a variety of applications, after being originally introduced as a shared ledger for the decentralized virtual currency Bitcoin (Nakamoto 2008). Such a decentralized blockchain is based on a peer-to-peer system, however, consensus rules applied in exactly the same manner at all peers define a protocol to ensure the integrity of transactions and their immutable recording in blocks. In this system, peers may act unpredictably (byzantine fault). Since transactions and blocks can be verified by all peers and the recording of any new block is carried out by an unpredictably selected peer through a mining process, consensus can still be achieved (Nakamoto 2008). In a public blockchain, anyone with sufficient resources can make transactions or take part in the mining process, usually by solving energy- and time-consuming computational problems (Proof of Work). In addition to various monetary implementations and time-stamped existence-proofs of records (Lemieux 2016), blockchain applications notably include smart contracts to formalize, program and execute contractual relationships in a cryptographically verifiable manner, initially introduced by Szabo (1997).

For IS literature, Notheisen et al. (2017) have provided an overview, distinguishing between the concepts *Blockchain technology*, *Trust-free economic systems*, *Bitcoin & crypto-currencies*, and *Financial service innovation & FinTech*. Concerning *Blockchain technology*, Brenig et al. (2016) conclude with a developed evaluation framework that decentralized systems in the case of Bitcoin can create economic value. Related to transactions for model creation in Section 3.2, Beck et al. (2016) demonstrate the trust-free property of blockchain-based systems, i.e. “economic transactions are guaranteed by the underlying blockchain”, thus being recorded immutable and cryptographically secured regarding their integrity. Greiner and Wang (2015) have addressed before the generalized trust-free systems approach.

Related to the representation of business processes and workflows on blockchains, Weber et al. (2016) and García-Bañuelos et al. (2017) provide an approach to implement reduced BPMN workflows as a smart contract, effectively using the Ethereum blockchain (Buterin et al. 2017) as a workflow engine. In contrast, this approach concerns the management of models when creating a process by multiple decentralized participants, where the actual execution may be off-loaded to a workflow-engine. Mendling et al. (2018) discuss research possibilities and refer to the business process management lifecycle. In the terminology of the authors, this approach relates to *Implementation* of “blockchain-based processes” by linking workflow tasks to a blockchain. It covers *Execution* aspects since blockchain “messages” in the form of transactions are made to a smart contract for linking versions to blocks and for agreement. Finally, *Monitoring* aspects are covered as the “global view of processes can be monitored independently by each involved party”. In addition, the example is similar to the use case (independently) chosen by this paper (Section 4), based on Fdhila et al. (2015).

Related to the immutable storage of enterprise models, the concept of a *Knowledge Blockchain* as a permissioned blockchain for model-based knowledge management has been demonstrated (Fill and Härer 2018). As opposed to the storage of enterprise models and model elements on a blockchain, this approach, independently developed as part of a thesis, is concerned with managing models of processes, workflows and instance states in a decentralize setting. However, it does not address permissions and existence proofs for model elements. It suggests how the collaborative creation of processes can be achieved by multiple decentralized participants across organizations without a coordinator, where the trust-free property is provided by a public blockchain. In this setting, block-size is a scarce resource and the time between blocks can impact performance. Therefore, meta-data needed to verify the integrity of models is stored on a blockchain instead of complete models (Section 3.2).

3 An approach for Decentralized Business Process Modeling and Instance Tracking

Building on the foundations described previously, this section details the main contributions regarding the creation of models for inter-organizational collaborative business processes and means to track instances. The overall architecture in Figure 1 functionally structures these aspects in terms of three main systems as Collaboration System, Instance Tracking System, and, Modeling System. The Modeling System defines a common language for all involved participants by setting modeling languages and file formats for each of the models for business process, workflows and instance states, where one model of a business process connects to n workflow models, which in turn connect to n instance state models for tracking instances. For collaborative modeling, the Collaboration System is applied for model management, with the integrity of the models being secured by a blockchain, and with agreements on specific model versions being formed by treating models as contracts. Based on agreed-upon versions, the Instance Tracking System is applied for tracking workflow instances through instance state models. Technologically, model management is based on a distributed version control system, while integrity and agreement functionality are realized by a blockchain.

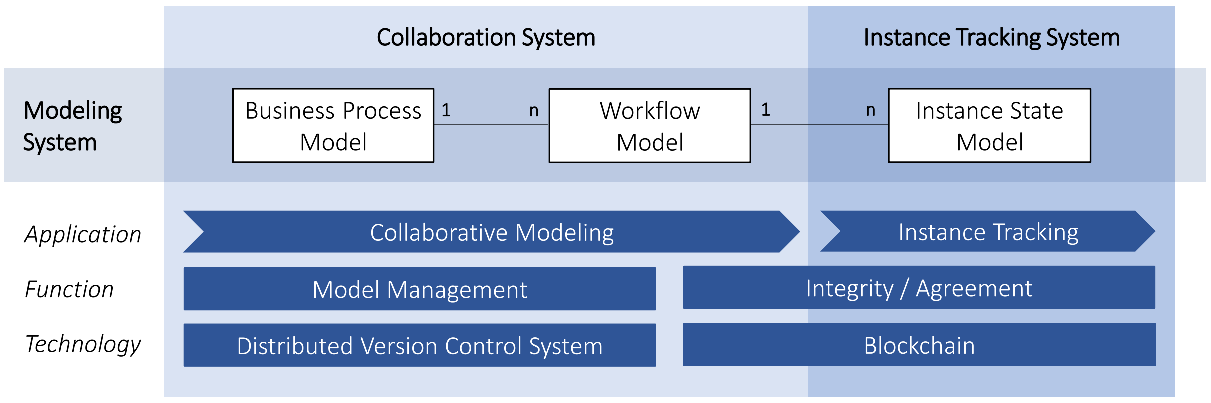


Figure 1. Overall architecture of the approach.

3.1 Modeling System

The Modeling System defines a common language between participants for describing the structural architecture of processes, their behavior using workflows, and instances in terms of the modeling languages. For defining the Modeling System at system build time, Figure 2 suggests the definition of modeling languages for business process models, workflow models, and instance state models with file formats. At system run time, model artifacts, i.e. files, are the basis for collaborative modeling during process build time with the Collaboration System described in Section 3.2 and instance tracking during process run time by means of the Instance Tracking System discussed in Section 3.3.

In a decentralized architecture, the interactions between peers are initially known by their structure, since any peer needs to identify others before any process activities take place. Therefore, peers are identified at the level of the business process model (Figure 2) by their structure first, comprising the structure of the decentralized architecture a peer is embedded in.

Following the network centric approach, the business process model represents a shared resource which is collaboratively created. In order to implement a transactional value creation corresponding to the architecture, n shared workflow models with inter-organizational activities have to be derived from the process model. For any shared workflow model, n peer workflow models can exist, since each peer has a private model (Fdhila et al. 2015) for its workflow, functioning as a local implementation of the shared workflow. For any shared workflow or peer workflow, n instance states may exist, modeled in the shared instance state model and peer instance model respectively.

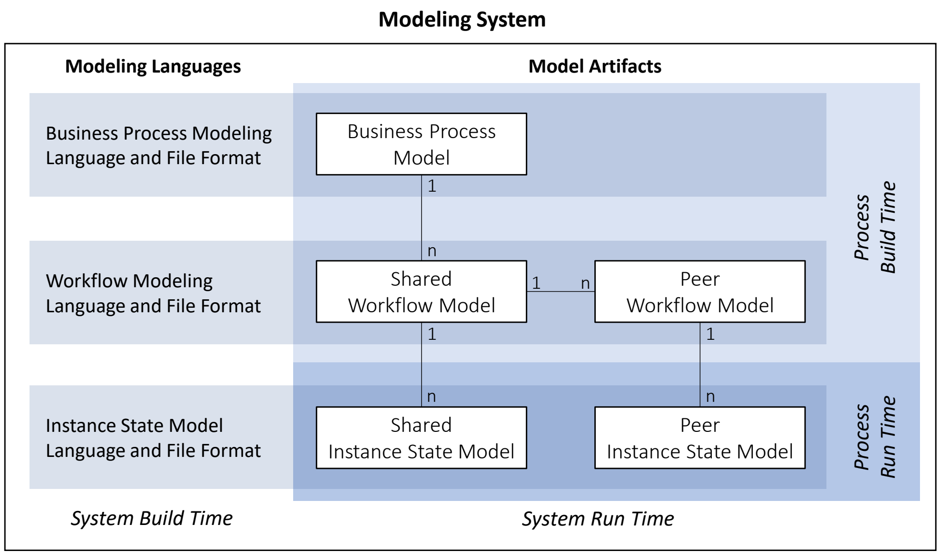

Modeling System
Modeling Languages
Model Artifacts
Business Process Modeling Language and File Format
Business Process Model
Workflow Modeling Language and File Format
Shared Workflow Model
Peer Workflow Model
Instance State Model Language and File Format
Shared Instance State Model
Peer Instance State Model
1
n
Process Build Time
Process Run Time
System Build Time
System Run Time

Figure 2. Modeling Languages and Model Artifacts to be specified.

The notion of shared workflows is similar to inter-organizational workflows as well as public and private processes (van der Aalst and Weske 2001), however, no particular modeling language is prescribed within the approach. The differentiation between business processes and workflows (e.g. Jablonski 1995) separates the modeling of individual business activities in the business process from their solution procedure (Pütz and Sinz 2010).

As modeling languages, the examples use the interaction schema (IAS) of the Semantic Object Model (SOM) (Ferstl and Sinz 2006, pp. 347-367) for modeling business processes, due to the lack of a concise structural diagram in BPMN (OMG 2014). On a workflow level, e.g. BPMN collaboration diagrams can be used to detail business process activities up to atomic tasks (OMG 2014, pp. 124, 132). Additionally, meta-model transformations between the levels of the modeling system can be employed by defining meta-models and relations as suggested by Sinz (1997, pp. 876-878) and the Model Driven Architecture (OMG 2003, pp. 5-2 and 5-3).

File formats depend on modeling languages as well as modeling tools used; in later examples, XML is employed. The procedure with simplistic models is illustrated in Figure 3. A model of the shared business process shows the process structure with interacting peers. Starting with the identification of peers, a process is aggregated, i.e. only the existence of peers is known at this point (P3 and P4 in Figure 3). Structural detailing decomposes process parts, e.g. for showing divisions of a supplier (lanes of P3 and P4).

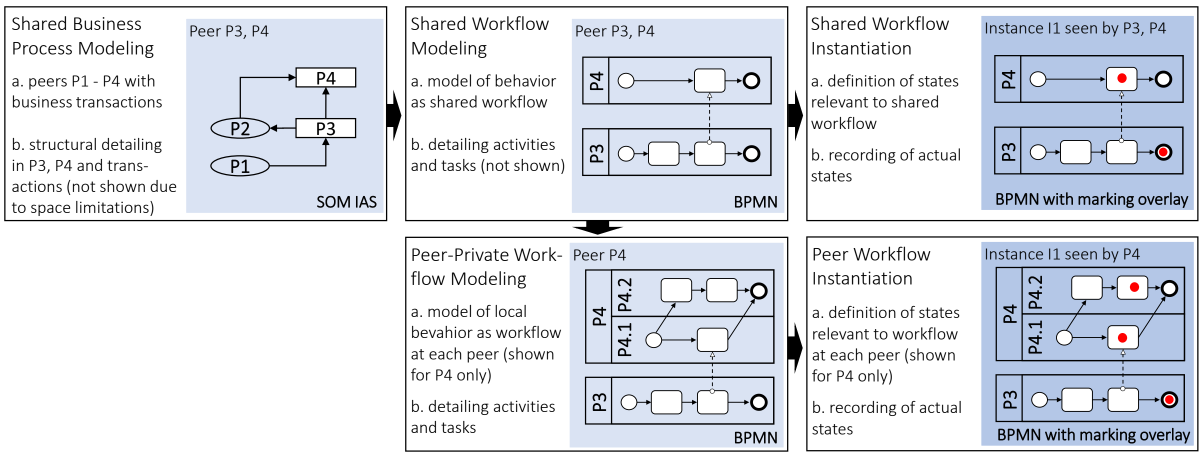

Shared Business Process Modeling
a. peers P1 - P4 with business transactions
b. structural detailing in P3, P4 and transactions (not shown due to space limitations)
Peer P3, P4
P1
P2
P3
P4
SOM IAS
Shared Workflow Modeling
a. model of behavior as shared workflow
b. detailing activities and tasks (not shown)
BPMN
Shared Workflow Instantiation
a. definition of states relevant to shared workflow
b. recording of actual states
Instance I1 seen by P3, P4
BPMN with marking overlay
Peer-Private Workflow Modeling
a. model of local bevahior as workflow at each peer (shown for P4 only)
b. detailing activities and tasks
Peer P4
P4.1
P4.2
Peer Workflow Instantiation
a. definition of states relevant to workflow at each peer (shown for P4 only)
b. recording of actual states
Instance I1 seen by P4

Figure 3. Modeling Procedure with example models.

Interactions between peers are modeled in SOM IAS notation as directed edges, representing business transactions (e.g. sell and deliver supplies), where the connected rectangular objects are the peers of the shared process, representing the objects of discourse, and ellipses are the surrounding peers not part of the shared process, representing environmental objects. On the workflow level, interactions are detailed to message flows with activities of the shared workflow. A private workflow is derived from a shared workflow such that the behavior of the shared workflow is implemented in higher detail in the private workflow, considering the properties discussed in Section 2.1. As an instance state model, this example uses BPMN collaboration diagrams overlaid with markings to represent a token present at a BPMN task of the shared and private instance state models. On this level, workflow instance states with relevance to the instantiation of the process are defined and actual instance states are recorded.

3.2 Collaboration System

The Collaboration System records the creation of models defined by the Modeling System from peers using a versioning concept, and conducts agreement procedures for agreeing on the specific models to use as a basis for forming contractual relationships. During process build time, possibly overlapping with process run time, inter-organizational processes with their workflows are shared and exchanged among participating peers using a Distributed Version Control System (DVCS), e.g. Git (Chacon and Straub 2014) and a smart contract based on the Ethereum blockchain (Buterin et al. 2017). In this hybrid system, the blockchain is deliberately limited to act as a layer for forming agreements and for providing integrity through trust-free transactions. Every participant is able to (automatically) verify the integrity of models and their immutability in order to provide consistent and dependable information to all participants and as a basis for informed agreements.

Model Artifacts can be managed by model versioning. A commit operation transitions Model Artifacts from one consistent state V_i of version i to another consistent state V_j of the subsequent version j in the sense of a database transaction with ACID properties. Operations required to submit transactions are commit, branch commit, and merge commit (Figure 4). With each commit, the newly persisted state represents a version with the associated Model Artifacts. When a modeler with domain knowledge makes changes, multiple modeling steps can be carried out in any number of models until a semantically correct state is reached across all models. Each individual model is required to be in a semantically and syntactically correct state (Bork and Sinz 2013, p. 31). When this is the case, a commit operation to a DVCS linked to a blockchain can be conducted.

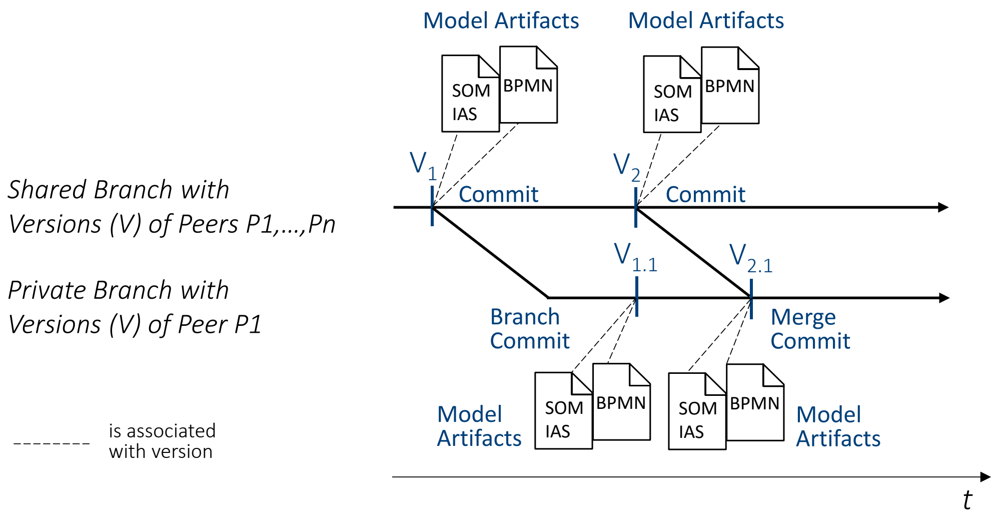


Figure 4. Versions (V) of shared and private Model Artifacts, created by commit operations.

A branch commit (Figure 4) creates a new branch from the chosen version as a private branch, private to a peer, or a shared branch, shared among n peers for collaborative modeling and instance tracking. The shared business process model and shared workflow models are jointly developed and detailed among n peers using a dedicated shared branch for P1,…, Pn (Figure 4). Each peer is able to model local workflows in a private branch, as shown for P1.

While collaborating on the Model Artifacts of shared process and workflow models in the shared branch, the local implementation in peer workflow models may be stored in a private branch only. An arbitrary number of shared branches and private branches can exist. A collaboration among n peers is initiated by the setup of a shared branch in a remotely accessible DVCS repository, or, by creating a shared branch from a private branch. From an existing collaboration forward, further branch commits are possible to initiate private branches and new collaborations, leading to any number of interleaving branches representing a network of connected processes. Possible combinations and characteristics are listed in Table 1.

Existing Branch	**Created Branch**	**Initiation Characteristic**
-	Private Branch	peer-private process initiation
-	Shared Branch	collaborative process initiation
Shared Branch	Private Branch	initiation of peer-private implementation of local process part
Shared Branch	Shared Branch	initiation of collaborative implementation of local process part
Private Branch	Private Branch	initiation of peer-private implementation of local process part
Private Branch	Shared Branch	collaborative process initiation from private process

Table 1. Initiation of collaborative processes and private processes by branch creation.

Figure 5 shows shared branches of one collaboration with private branches at process build time. Model Artifacts are associated with each version (Figure 4), as previously described with a. shared business process models and shared workflow models at the shared branch and b. private workflow models at the private branch. In order to use the shared branch as a basis for agreements and contracts, meta-data of versions can be mirrored on the blockchain (on-chain) in a block mined after a commit operation by the committing peer. While in the DVCS a branch contains Model Artifacts, visible to its peers, it is sufficient for the public blockchain to only contain the structure shown in Figure 6 with meta-data and its Hash value of each version, recorded by a smart contract in individual blocks. For meta-data, the version number, the committing peer (address), the hash value of models calculated as hash function H(Model_Artifacts), and the agreement procedure results are sufficient. The agreement procedure uses a voting mechanism to signal agreement for or against using a committed Model Artifact as a contract.

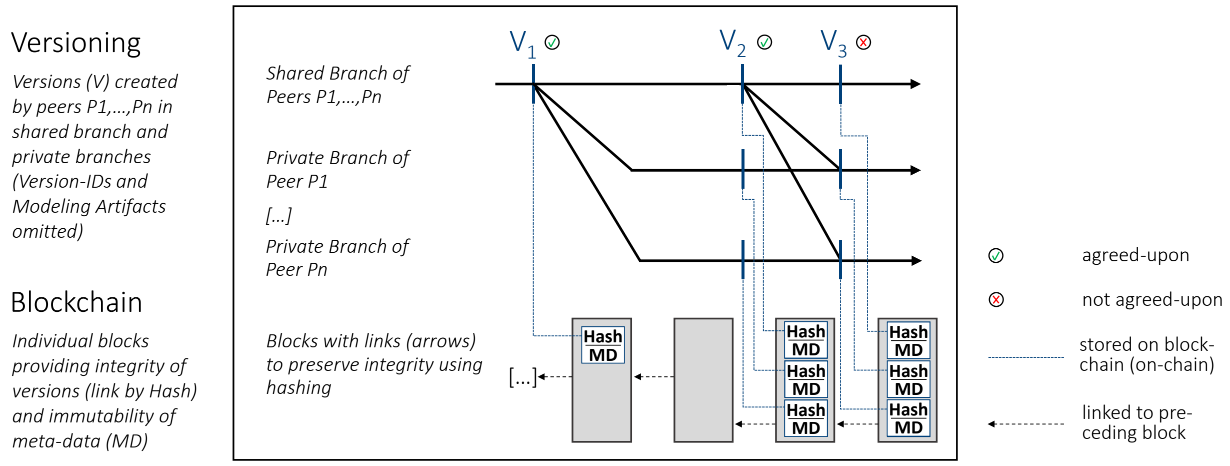

Versioning
Versions (V) created by peers P1,…,Pn in shared branch and private branches (Version-IDs and Modeling Artifacts omitted)
Blockchain
Individual blocks providing integrity of versions (link by Hash) and immutability of meta-data (MD)
Shared Branch of Peers P1,…,Pn
Private Branch of Peer P1
[…]
Private Branch of Peer Pn
Blocks with links (arrows) to preserve integrity using hashing
V1
V2
V3
[…]
Hash
MD
agreed-upon
not agreed-upon
stored on block-chain (on-chain)
linked to pre-ceding block

Figure 5. Versions (V) linked to a blockchain for integrity and agreement.

Contracting participants require a unanimous agreement, by vote in the fashion of a two-phase commit protocol (van Steen and Tanenbaum 2017, pp. 483-488), resulting in an agreed-upon model (check-mark in Figure 5) in this case. In all other cases, the model is not agreed-upon (cross-mark in Figure

5). An agreement states the commitment to use and implement a model, representing a contract. Using models as contracts therefore does not directly refer to legal contracts, however, it may outline contractual relationships, formulated as precise as the domain requires through a domain-specific model. Only in case participants agree on a model, they need to merge it. The agreement procedure can be carried out for any version on a shared branch, with peers signaling by calling a smart contract function, which solely stores meta-data (MD) and Hash (Figure 5) on-chain. The concept of an address in Ethereum can be used for separating smart contracts, e.g. corresponding to separate private branches. Hash and MD function as links to versions and models, where Hash is calculated using a hash function H(MD) = Hash (e.g. using SHA-256 for H). It follows that the public blockchain does not contain models of the DVCS, while the integrity of models and agreements is secured by the blockchain.

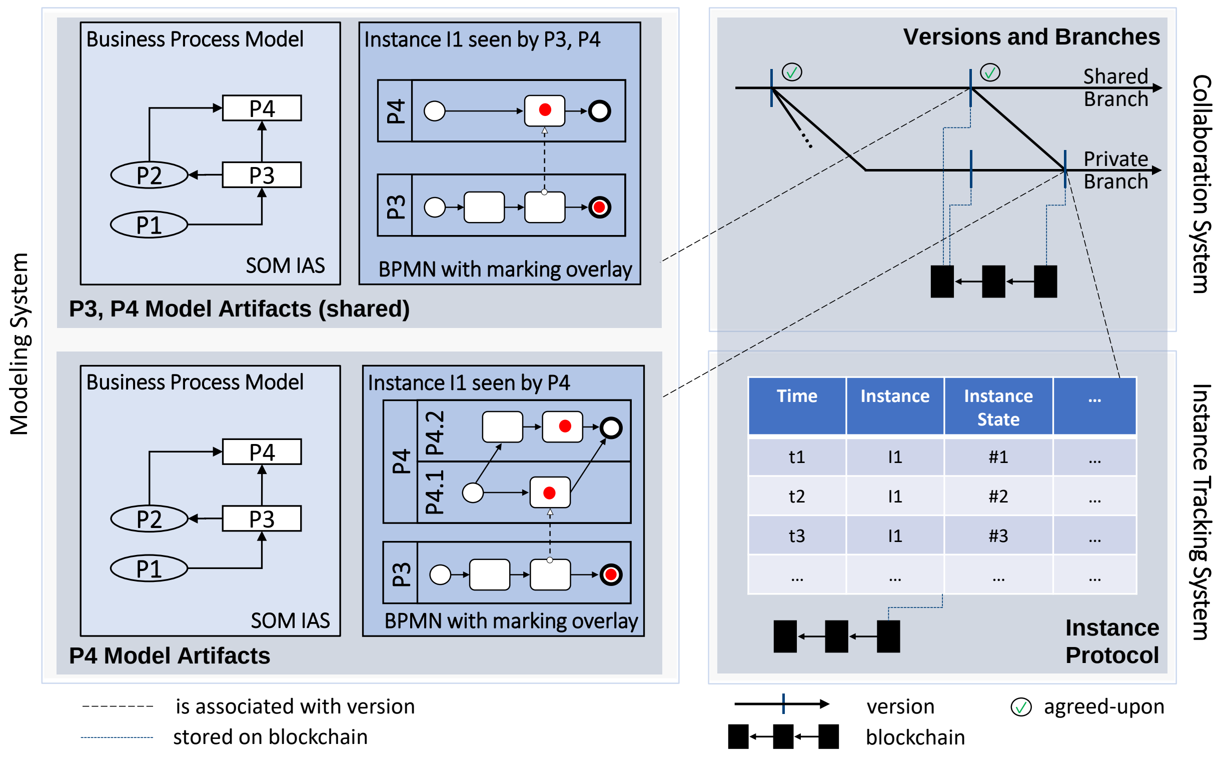
3.3 Instance Tracking System

The Instance Tracking System distributes models of workflow instance states among peers. The discussed trust-free property and the integrity property apply to instance states as well, if they are secured by a blockchain. Analogously to the Collaboration System, handling meta-data on-chain is sufficient for integrity to depend on, while models do not need to be publicly stored. For storing instance states with relevance to the workflow execution, states of the shared workflow and the public workflow can be separated using branching as suggested before. Therefore, the instance tracking system relies on meta-data of models, present on the blockchain, as well as actual models, present in the DVCS. Instance state models represent specific instance states which are of relevance to the workflow execution, e.g. the initial and final states of the process, as well as intermediate states important for collaboration, e.g. when a supplier delivers a supply part which now needs to be handled by another manufacturer. This allows for relevant states to be defined at any point in time and checked against actual instances, in a trust-free way, to the extent of the integrity-providing blockchain records. States to be checked against are referred to as check-point states, whereas actually occurring states are referred to as actual states. When instantiating an agreed-upon version of a shared or private workflow, the according instance state models are part of the respective shared and private branches. Shared models are augmented by private models, since they show instance states in the private implementation of the shared workflow (Section 3.2). In the same way a shared workflow can be a basis for a contractual relationship in case of an agreement, an instance state model can be used to record whether the instantiation occurred according to the agreed-upon workflow model. In a collaboration with a supplier, it might be useful to see an up-to-date state of the shared workflow, where e.g. a quality check has been or has not been performed yet. In a co-opetition setting, it might be useful for a company to get selective insights into a competitor's workflow, who is collaborating in certain activities or tasks.

On a technical level, the comparison of actual states and check-point states can be accomplished using hash values. In combination with the Collaboration System, instance states of specific versions can be shown for shared and private workflows.

Figure 6 shows an Instance I1 for participants P3, P4 as a shared model (upper left) and for P4 as a private model (lower left). The Model Artifacts contain an execution state as indicated by the BPMN notation with marking overlay, where only P4 sees the private workflows, which implement the shared workflows. These Model Artifacts are associated with versions (upper right), agreed-upon in the shared branch (check-mark), merged into the private branch. For Instance I1, an Instance Protocol (lower right) can be derived, to show the actual instance states over time. On the blockchain, the integrity of versions and branches is secured, as indicated by the dotted lines, by means of meta-data and hashing. Additionally, the Instance Protocol is stored on-chain, to provide instance state information in a trust-free way.

Regarding the derivation of the Instance Protocol, the order of actual states for each shared or private branch on a model level can be determined. Since it is always known how instances are embedded in branches, i.e. where they are derived from, only the structure of the workflow is required to track instantiation. For this reason, a partial or total ordering of workflow activities is not required in models, since this model-level-approach relies on a modeler defining and confirming instance states with correct semantics. With information from the instance states models, an instantiation protocol can summarize the total order of actual states on a model level.



Time	Instance	Instance State	...
t1	I1	#1	...
t2	I1	#2	...
t3	I1	#3	...
...	...	...	...

Figure 6. Modeling System, Collaboration System and Instance Tracking System for Instance I1.

4 Implementation and Use Case

This section demonstrates the applicability of the approach within an implemented software prototype and discusses results regarding research questions 1 and 2. First, a use case to conduct the evaluation is introduced. As a representation of a prototypical collaborative business process with decentralized participants, a production-oriented business process by Fdhila et al. (2015) was chosen. Secondly, the use case is modeled by means of the suggested approach. Business process models, derived workflow models and models of workflow instances are managed by the implemented software prototype. Finally, conclusions regarding research questions 1 and 2 are drawn.

4.1 Use Case

The use case by Fdhila et al. (2015) models the supply chain of a manufacturing process, where multiple process changes originate from different participants. The participants *Manufacturer*, *Middleman*, *Special Carrier*, and a *Supplier A* carry out a collaborative business process for manufacturing a product for *Bulk Buyer*, requiring an “Intermediate A” from *Supplier A* and an “Intermediate B” from *Supplier B*. The participants of the shared process do not know the processes and workflows of *Bulk Buyer* and *Supplier B*, but only their interactions in the form of business transactions on a business process level and message flows on a workflow level. Five changes to the process are made in this scenario (Fdhila et al. 2015, pp. 2 and 5). In δ1, a task for quality testing the product is removed from the workflow of *Manufacturer*, while in δ2 a message flow with a quality report from *Supplier A* to *Manufacturer* is added as compensation. δ3 introduces a private task for a quick quality test at *Manufacturer*. δ4 adds a full quality test at *Supplier A* as a private task, prior to the quality report. In δ5, a compliance rule is formulated outside of the workflow model.

4.2 Modeling and Management of Models in Software Prototype

The identified peers and their interactions were detailed in a structural view using a SOM Interaction Schema (IAS) for laying out the business transactions. For representing the behavior in workflow models, BPMN collaboration diagrams were derived from the process model. Languages and file formats of the Modeling System are therefore 1) SOM for the business process model; stored in custom XML files or ADL files created by a SOM tool based on ADOxx (Ferstl et al. 2016). 2) BPMN 2.0 for the shared and private workflows with the standardized BPMN 2.0 XML file format (OMG 2014, pp. 138-142) as created by the employed bpmn-js library (Camunda 2017). Finally, 3) an extension of BPMN for overlaying instance states in instance state models using a developed java-script extension for bpmn-js, stored as BPMN 2.0 XML file with extension attributes. The prototype[1] is implemented in Java SE 9 and uses Git as DVCS; for linking versions to a public blockchain, a smart contract has been implemented on the Ethereum blockchain.

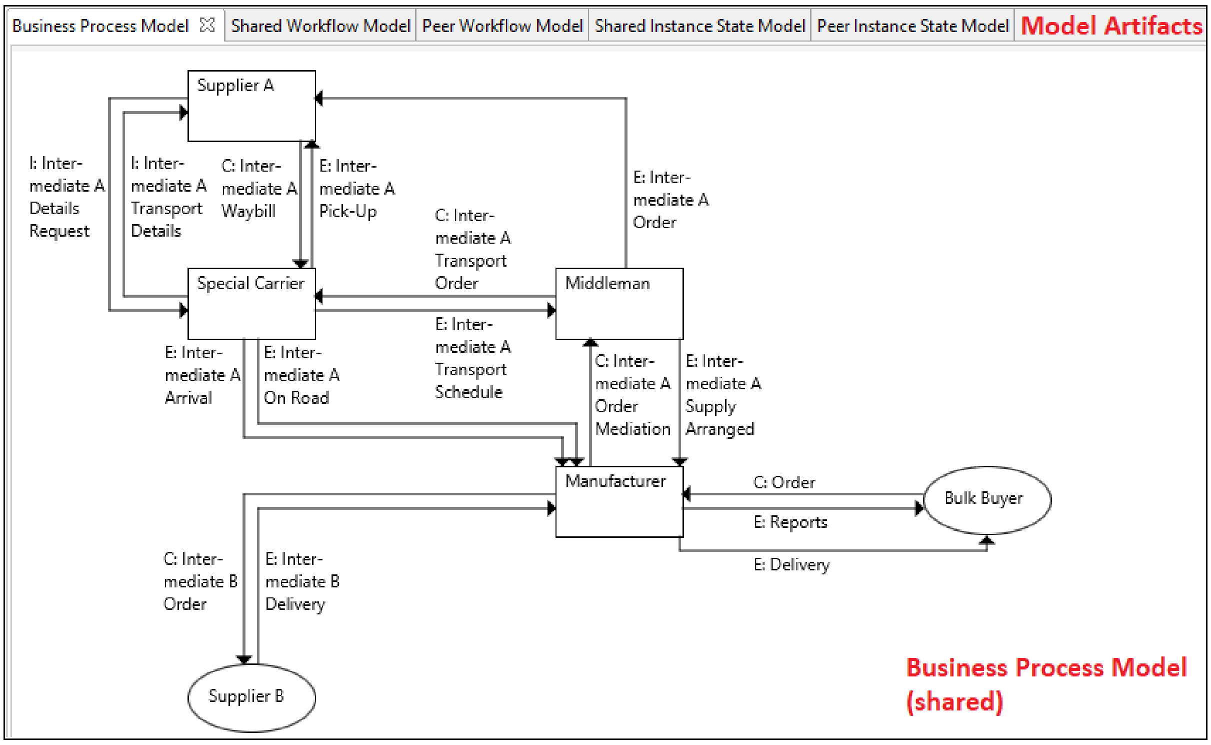

Business Process Model
Shared Workflow Model
Peer Workflow Model
Shared Instance State Model
Peer Instance State Model
Model Artifacts
Supplier A
I: Inter-mediate A Details Request
I: Inter-mediate A Transport Details
C: Inter-mediate A Waybill
E: Inter-mediate A Pick-Up
C: Inter-mediate A Transport Order
E: Inter-mediate A Order
Special Carrier
Middleman
E: Inter-mediate A Transport Schedule
E: Inter-mediate A Arrival
E: Inter-mediate A On Road
C: Inter-mediate A Order Mediation
E: Inter-mediate A Supply Arranged
Manufacturer
C: Order
E: Reports
E: Delivery
Bulk Buyer
C: Inter-mediate B Order
E: Inter-mediate B Delivery
Supplier B
Business Process Model (shared)

Figure 7. Structure of Collaborative Business Process in SOM IAS notation in software tool.

Figure 7 gives the structural overview of the business process after peer identification and detailing with participants either as part of the shared process (rectangles) or not as part of the shared process (ellipses), connected by business transactions to initiate (I), contract (C), or enforce (E). Following the derivation of a shared workflow as defined by the case study, private workflows in according branches for each peer were derived, including private tasks, not visible to other peers. Ongoing from the initial version, four changes of the change scenario could be implemented, with changes δ1 and δ2 made to the shared workflow and changes δ3 and δ4 made to the private workflows of *Manufacturer* and *Supplier A* respectively. Change δ5 could not be implemented, since compliance rules are not expressible by the domain-specific languages chosen in the Modeling System; for adding according Model Artifacts, a specification of an appropriate language is required at system build time.

[1] https://github.com/fhaer/Decentralized-Process-Modeling-and-Instance-Tracking

A shared workflow model is shown in Figure 8 on the right, while the private workflow of *Manufacturer* can be seen on the left. In addition, tracked instances are shown with markings (red) for executed tasks. Changes to the workflows are tracked in different versions (lower middle). The agreement procedure for Version V2 has been completed with no agreed-upon model, while in V3, peers agreed to use and implement the changed (δ1 and δ2) model. The change δ3, local and private to *Manufacturer*, is visible in the task "Quick Test Intermediate A*"* (left model), as part of the instantiation of V3.Ma.1 in instance I1, while δ4 is outside the frame (*Supplier A*). In the shared instance state model (right model), δ3 is not visible.

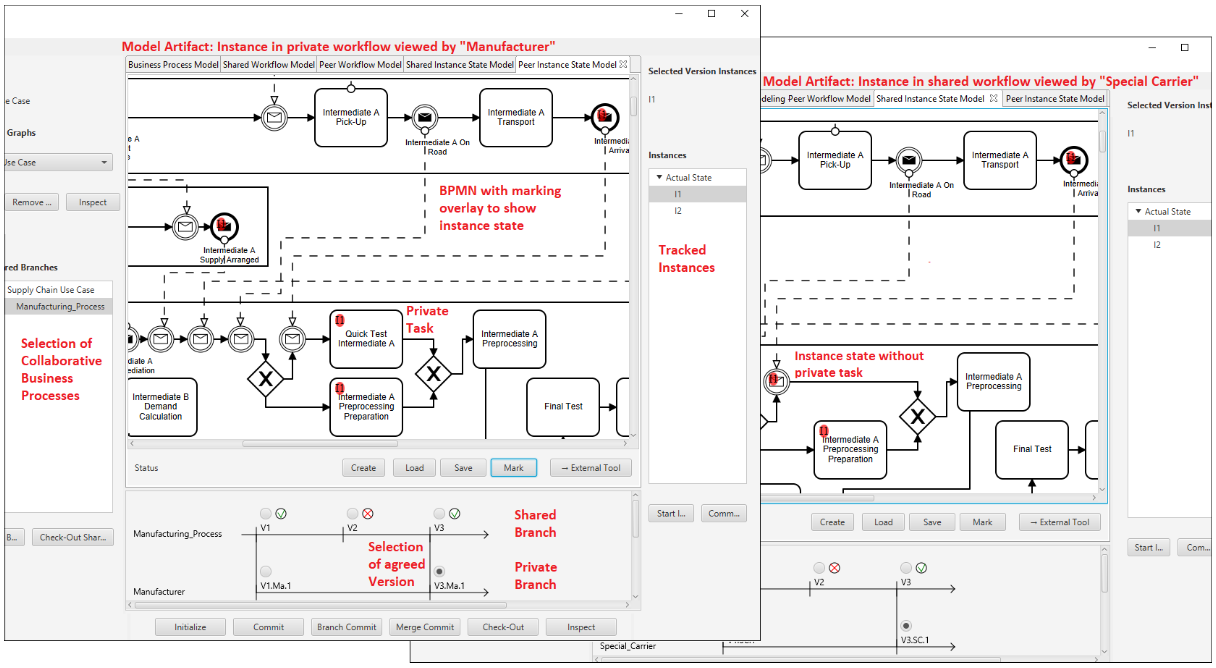

Model Artifact: Instance in private workflow viewed by "Manufacturer"
Model Artifact: Instance in shared workflow viewed by "Special Carrier"
BPMN with marking overlay to show instance state
Tracked Instances
Selection of Collaborative Business Processes
Private Task
Instance state without private task
Shared Branch
Private Branch
Selection of agreed Version
Intermediate A Pick-Up
Intermediate A Transport
Intermediate A On Road
Intermediate A Supply Arranged
Quick Test Intermediate A
Intermediate A Preprocessing
Intermediate A Preprocessing Preparation
Intermediate B Demand Calculation
Final Test
Selected Version Instances
Instances
Actual State
I1
I2
Manufacturing_Process
Manufacturer
Special_Carrier
V1
V2
V3
V1.Ma.1
V3.Ma.1
V3.SC.1
Create
Load
Save
Mark
→ External Tool
Initialize
Commit
Branch Commit
Merge Commit
Check-Out
Inspect

Figure 8. *Prototype implementation with Instance Tracking at Manufacturer, Special Carrier.*

4.3 Discussion of results

Regarding the aspects of collaborative processes (Section 2.1), models for a choreography and private models are established in this use case. Models of the choreography are considered to be public and shared among participants; they are therefore part of the private models viewed by participants (Figure 8). The suggested *approach* is top down, starting from a structural overview as SOM IAS. Partner selection can be characterized as *static* in this use case, however, the versioning approach also allows for *dynamic* selection, since changes in the business process can include new peers, with the creation of corresponding branches and smart contracts for agreement procedures. Participants are responsible for implementing local models in accordance to shared models they agreed to, for realizing a collaboration. Due to the choice of handling common semi-formal notations for enterprise models, the *properties* (Section 2.1) for *consistency*, *compatibility*, and *realizability* are not guaranteed by this approach. For example, BPMN with relaxed soundness (Dehnert and Rittgen 2001) limits the scope of possible guarantees, following the rationale of using semi-formal models discussed in Section 3.1 and 1.

With the use case discussed, the suggested approach addresses both research questions as follows.

Research question 1: By running the software prototype at each participant, modeling is carried out from a local perspective. Shared models are collaboratively created using model versioning with a DVCS and differentiate between public and private workflows. In order to achieve consensus on models, agreement procedures are conducted by voting on versions. For implementing decentralization, the integrity of models is secured by a public blockchain. Since participants can check the integrity of models on their own, participants can rely on models and do not need a coordinator. As a result, the suggested approach represents one possibility for answering research question 1 by addressing model creation, collaborative modeling, forming agreements, and decentralization.

Research question 2: While the instantiation itself is not a part of the approach, instance states with relevance to the workflow can be tracked. Models of instance states represent processes at run time for each participant using markings. The decentralization aspect is addressed by the methods established previously, by linking versions of instance state models to a public blockchain. In conclusion, research question 2 can be answered partially, as participants can only model and track some instance states with relevance to the process. This technical limitation is imposed by using a public blockchain, which limits block size and sets the time interval between blocks.

5 Conclusion and Next Steps

The recombination of existing model versioning and blockchain technologies allow for the collaborative creation of agreed-upon enterprise models in a network of decentralized peers, as well as the tracking of process instances using models with meta-data. Core components of this approach are the *Modeling System*, *Collaboration System*, and *Instance Tracking System*. The *Modeling System* describes a common language for modeling at the levels of business processes, workflows, and instance states, where workflows and instance states may be shared or private to a peer. The approach does not depend on specific modeling languages, as it provides versioning mechanisms for model artifacts, i.e. files, to be defined for each modeling language as part of the *Modeling System*. The *Collaboration System* concerns process build time, where versions of model artifacts are stored in a version control system, linked to a permissionless blockchain as an agreement layer providing trust-free transactions. Agreements are formed using a voting-mechanism in order to agree on models, which may be used by participants as a basis for contractual agreements. The *Instance Tracking System* is employed at process run time for recording models of instance states in the version control system. According meta-data is placed on the blockchain in order to provide means for tracking process instances without storing models on the public blockchain. A use case evaluates the approach with regard to a collaborative business process. According business process and workflow models are demonstrated in an implemented software tool, with limitations concerning the extent of modeling languages, their required specification at build time, and possible guarantees of collaborative business process properties. The analysis of the limits to the enforceability of collaborative business properties with semi-formal models, e.g. through formalized constraints, remains to be investigated. Research regarding the integration with a meta-modeling platform is planned for evolving towards generic integrity-providing blockchain applications based on meta-models.

References

Beck, R., J. S. Czepluch, N. Lollike and S. Malone (2016). "Blockchain - the Gateway to Trust-Free Cryptographic Transactions Turkey, June 12-15, 2016". In: *24th European Conference on Information Systems, ECIS 2016, Istanbul, Turkey, June 12-15, 2016,* Research Paper 153. URL: http://aisel.aisnet.org/ecis2016_rp/153 (visited on 11/12/2017).

Bork, D. and H.-G. Fill (2014). "Formal Aspects of Enterprise Modeling Methods. A Comparison Framework Waikoloa, HI, USA, January 6-9, 2014". In: *47th Hawaii International Conference on System Sciences, HICSS 2014, Waikoloa, HI, USA, January 6-9, 2014*: IEEE Computer Society, pp. 3400–3409. URL: https://doi.org/10.1109/HICSS.2014.422 (visited on 11/12/2017).

Bork, D. and E.-J. Sinz (2013): „Bridging the Gap From a Multi-View Modelling Method to the Design of a Multi-View Modelling Tool" *Enterprise Modelling and Information Systems Architectures - An International Journal* 8 (2), 25-41.

Brenig, C., J. Schwarz and N. Rückeshäuser (2016). "Value of Decentralized consensus Systems - Evaluation Framework Turkey, June 12-15, 2016". In: *24th European Conference on Information Systems, ECIS 2016, Istanbul, Turkey, June 12-15, 2016,* Research Paper 75. URL: http://aisel.aisnet.org/ecis2016_rp/75 (visited on 11/12/2017).

Brosch, P., G. Kappel, P. Langer, M. Seidl, K. Wieland and M. Wimmer (2012). "An Introduction to Model Versioning". In: *Proceedings of the 12th International Conference on Formal Methods for the Design of Computer, Communication, and Software Systems: Formal Methods for Model-driven Engineering*. Berlin, Heidelberg: Springer-Verlag, pp. 336–398. URL: http://dx.doi.org/10.1007/978-3-642-30982-3_10 (visited on 11/12/2017).

Buterin, V., G. Wood, J. Wilcke, Committers (2017). *Ethereum White Paper*. *A Next-Generation Smart Contract and Decentralized Application Platform*. Technical Report. URL: https://github.com/ethereum/wiki/wiki/White-Paper (visited on 11/12/2017).

Camunda (2017). *bpmn-js*. *A Quick Introduction*. Technical Report. URL: https://bpmn.io/toolkit/bpmn-js/walkthrough/ (visited on 11/12/2017).

Chacon, S. and B. Straub (2014). *Pro Git*. Second edition. New York: Apress.

Decker, G. and M. Weske (2007). "Behavioral Consistency for B2B Process Integration CAiSE 2007, Trondheim, Norway, June 11-15, 2007, Proceedings". In: *Advanced Information Systems Engineering, 19th International Conference, CAiSE 2007, Trondheim, Norway, June 11-15, 2007, Proceedings*. Ed. by J. Krogstie, A. L. Opdahl, G. Sindre: Springer, pp. 81–95. URL: https://doi.org/10.1007/978-3-540-72988-4_7 (visited on 11/12/2017).

Dehnert, J. and P. Rittgen (2001). "Relaxed Soundness of Business Processes". In K. R. Dittrich, A. Geppert and M. C. Norrie (eds.) *Advanced Information Systems Engineering: 13th International Conference, CAiSE 2001 Interlaken, Switzerland, June 4-8, 2001 Proceedings*, pp. 157–170. Berlin, Heidelberg: Springer Berlin Heidelberg. URL: https://doi.org/10.1007/3-540-45341-5_11 (visited on 11/12/2017).

Derntl, M., P. Nicolaescu, S. Erdtmann, R. Klamma and M. Jarke (2015). "Near Real-Time Collaborative Conceptual Modeling on the Web". In P. Johannesson, M. L. Lee, S. W. Liddle, A. L. Opdahl and Ó. Pastor López (eds.) *Conceptual Modeling: 34th International Conference, ER 2015, Stockholm, Sweden, October 19-22, 2015, Proceedings*, pp. 344–357. Cham: Springer International Publishing.

Dollmann, T., C. Houy, P. Fettke and P. Loos (2011). "Collaborative Business Process Modeling with CoMoMod - A Toolkit for Model Integration in Distributed Cooperation Environments". In: *2011 IEEE 20th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*: IEEE, pp. 217–222.

Fdhila, W., S. Rinderle-Ma, D. Knuplesch and M. Reichert (2015). "Change and Compliance in Collaborative Processes 2015, New York City, NY, USA, June 27 - July 2, 2015". In: *2015 IEEE International Conference on Services Computing, SCC 2015, New York City, NY, USA, June 27 - July*

2, 2015: IEEE Computer Society, pp. 162–169. URL: https://doi.org/10.1109/SCC.2015.31 (visited on 11/12/2017).

Ferdinand, J.-P., U. Petschow and S. Dickel (eds.) (2016). *The Decentralized and Networked Future of Value Creation.* Cham: Springer International Publishing.

Ferstl, O. K. and E. J. Sinz (2006). "Modeling of Business Systems Using SOM". In P. Bernus, K. Mertins and G. Schmidt (eds.) *Handbook on Architectures of Information Systems*, pp. 347–367. Berlin, Heidelberg: Springer Berlin Heidelberg. URL: https://doi.org/10.1007/3-540-26661-5_15 (visited on 11/12/2017).

Ferstl, O. K., E. J. Sinz and D. Bork (2016). "Tool Support for the Semantic Object Model". In D. Karagiannis, H. C. Mayr and J. Mylopoulos (eds.) *Domain-Specific Conceptual Modeling*, pp. 291–310. Berlin: Springer.

Fill, H.-G. and D. Karagiannis (2013). "On the Conceptualisation of Modelling Methods Using the ADOxx Meta Modelling Platform" *Enterprise Modelling and Information Systems Architectures* 8 (1), 4–25.

Fill, H.-G. and F. Härer (2018). "Knowledge Blockchains: Applying Blockchain Technologies to Enterprise Modeling". In: *51st Hawaii International Conference on System Sciences, HICSS 2018, Hilton Waikoloa Village, Hawaii, USA, January 3-6, 2018*. URL: http://hdl.handle.net/10125/50398 (visited 18/04/03).

Fröwis, M. and R. Böhme (2017). "In Code We Trust?". In J. Garcia-Alfaro, G. Navarro-Arribas, H. Hartenstein and J. Herrera-Joancomartí (eds.) *Data Privacy Management, Cryptocurrencies and Blockchain Technology: ESORICS 2017 International Workshops, DPM 2017 and CBT 2017, Oslo, Norway, September 14-15, 2017, Proceedings*, pp. 357–372. Cham: Springer International Publishing.

García-Bañuelos, L., A. Ponomarev, M. Dumas and I. Weber (2017). "Optimized Execution of Business Processes on Blockchain 2017, Barcelona, Spain, September 10-15, 2017, Proceedings". In: *Business Process Management - 15th International Conference, BPM 2017, Barcelona, Spain, September 10-15, 2017, Proceedings*. Ed. by J. Carmona, G. Engels, A. Kumar: Springer, pp. 130–146. URL: https://doi.org/10.1007/978-3-319-65000-5_8 (visited on 11/12/2017).

Geiger, M., S. Harrer, J. Lenhard and G. Wirtz (2017). "BPMN 2.0. The state of support and implementation" *Future Generation Computer Systems*.

Greiner, M. E. and H. Wang (2015). "Trust-free Systems - a New Research and Design Direction to Handle Trust-Issues in P2P Systems. The Case of Bitcoin Rico, August 13-15, 2015". In: *21st Americas Conference on Information Systems, AMCIS 2015, Puerto Rico, August 13-15, 2015*: Association for Information Systems.

Hewelt, M. and M. Weske (2016). "A Hybrid Approach for Flexible Case Modeling and Execution Brazil, September 18-22, 2016, Proceedings". In: *Business Process Management Forum - BPM Forum 2016, Rio de Janeiro, Brazil, September 18-22, 2016, Proceedings*. Ed. by M. La Rosa, P. Loos, O. Pastor: Springer, pp. 38–54.

Hofreiter, B. and C. Huemer (2008). "A model-driven top-down approach to inter-organizational systems. From global choreography models to executable BPEL". In: *IEEE Joint Conference on E-Commerce Technology (CEC'08) and Enterprise Computing, E-Commerce, and E-Services (EEE'08)*: IEEE Computer Society, pp. 136–145. URL: http://publik.tuwien.ac.at/files/PubDat_166333.pdf (visited on 11/12/2017).

Jablonski, S. (1995). "On the Complementarity of Workflow Management and Business Process Modeling" *SIGOIS Bull* 16 (1), 33–38.

Karagiannis, D., H. C. Mayr and J. Mylopoulos (2016). *Domain-Specific Conceptual Modeling*. Berlin: Springer.

Kurose, J. F. and K. W. Ross (2017). *Computer Networking. A Top-Down Approach.* 7th Edition. Boston, MA: Pearson.

Lemieux, V. L. (2016). "Trusting records. Is Blockchain technology the answer?" *Records Management Journal* 26 (2), 110–139.

Liu, S., Y. Zheng, H. Shen, S. Xia and C. Sun (2006). “Real-Time Collaborative Software Modeling Using UML with Rational Software Architect”. In: *2006 International Conference on Collaborative Computing: Networking, Applications and Worksharing*, pp. 1–9.

Maróti, M., T. Kecskés, R. Kereskényi, B. Broll, P. Völgyesi, L. Jurácz, T. Levendovszky and Á. Lédeczi (2014). “Next Generation (Meta)Modeling. Web- and Cloud-based Collaborative Tool Infrastructure”. In: *MPM@MoDELS: CEUR-WS.org Vol. 1237*. URL: http://ceur-ws.org/Vol-1237/paper5.pdf (visited on 11/27/2017).

Mendling, J., I. Weber, W. M. P. van der Aalst, J. Vom Brocke, C. Cabanillas, F. Daniel, S. Debois, C. Di Ciccio, M. Dumas, S. Dustdar, A. Gal, L. Garc-Bañuelos, G. Governatori, R. Hull, M. La Rosa, H. Leopold, F. Leymann, J. Recker, M. Reichert, H. A. Reijers, S. Rinderle-Ma, A. Solti, M. Rosemann, S. Schulte, M. P. Singh, T. Slaats, M. Staples, B. Weber, M. Weidlich, M. Weske, X. Xu and L. Zhu (2018). “Blockchains for Business Process Management - Challenges and Opportunities” *ACM Trans. Management Inf. Syst.* 9 (1), 4:1-4:16.

Nakamoto, S. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. Technical Report. URL: https://bitcoin.org/bitcoin.pdf (visited on 11/12/2017).

Nærland, K., C. Müller-Bloch, R. Beck and S. Palmund (2017). “Blockchain to Rule the Waves - Nascent Design Principles for Reducing Risk and Uncertainty in Decentralized Environments - Transforming Society with Digital Innovation, ICIS 2017, Seoul, South Korea, December 10-13, 2017”. In: *Proceedings of the International Conference on Information Systems - Transforming Society with Digital Innovation, ICIS 2017, Seoul, South Korea, December 10-13, 2017*. Ed. by Y. J. Kim, R. Agarwal, J. K. Lee: Association for Information Systems.

Nicolaescu, P., M. Rosenstengel, M. Derntl, R. Klamma and M. Jarke (2017). “Near real-time collaborative modeling for view-based Web information systems engineering” *Information Systems*. URL: http://dx.doi.org/10.1016/j.is.2017.07.008 (visited on 11/27/2017).

Nicoletti, B. (2018). “The Future. Procurement 4.0”. In *Agile Procurement : Volume II: Designing and Implementing a Digital Transformation*, pp. 189–230. Cham: Springer International Publishing. URL: https://doi.org/10.1007/978-3-319-61085-6_8.

Notheisen, B., F. Hawlitschek and C. Weinhardt (2017). “Breaking down the Blockchain Hype - towards a Blockchain Market Engineering Approach Portugal, June 5-10, 2017”. In: *25th European Conference on Information Systems, ECIS 2017, Guimarães, Portugal, June 5-10, 2017*. Ed. by I. Ramos, V. Tuunainen, H. Krcmar. URL: http://aisel.aisnet.org/ecis2017_rp/69 (visited on 11/12/2017).

Object Management Group (OMG) (2003). *MDA Guide Version 1.0.1* (omg/2003-06-01). URL: http://www.omg.org/cgi-bin/doc?omg/03-06-01.pdf (visited on 11/12/2017).

Object Management Group (OMG) (2014). *Business Process Model and Notation (BPMN) Version 2.0.2* (formal/2013-12-09). URL: http://www.omg.org/spec/BPMN/2.0.2 (visited on 11/12/2017).

Pütz, C. and E. J. Sinz (2010). “Model-driven Derivation of BPMN Workflow Schemata from SOM Business Process Models” *Enterprise Modelling and Information Systems Architectures* 5 (2), 57–72.

Ryu, K. and E. Yücesan (2007). “CPM. A collaborative process modeling for cooperative manufacturers” *Advanced Engineering Informatics* 21 (2), 231–239.

Sinz, E. J. (1997). “Architektur betrieblicher Informationssysteme” (in german). In P. Rechenberg and G. Pomberger (eds.) *Informatik-Handbuch*. München: Hanser.

Steinmetz, R. and K. Wehrle (2005). *Peer-to-peer systems and applications.* Berlin: Springer.

Sun, C. and R. Sosič (1999). “Optimal Locking Integrated with Operational Transformation in Distributed Real-time Group Editors”. In: *Proceedings of the Eighteenth Annual ACM Symposium on Principles of Distributed Computing*. New York, NY, USA: ACM, pp. 43–52.

Szabo, N. (1997). “Formalizing and Securing Relationships on Public Networks” *First Monday* 2 (9). URL: http://dx.doi.org/10.5210/fm.v2i9.548 (visited on 11/27/2017).

van der Aalst, W. M. P., K. M. van Hee, A. H. M. ter Hofstede, N. Sidorova, H. M. W. Verbeek, M. Voorhoeve and M. T. Wynn (2011). “Soundness of workflow nets. Classification, decidability, and analysis” *Formal Aspects of Computing* 23 (3), 333–363.

van der Aalst, W. M. P. and M. Weske (2001). “The P2P Approach to Interorganizational Workflows”. In G. Goos, J. Hartmanis, J. van Leeuwen, K. R. Dittrich, A. Geppert and M. C. Norrie (eds.) *Advanced Information Systems Engineering*, pp. 140–156. Berlin, Heidelberg: Springer Berlin Heidelberg.

van Steen, M. and A. S. Tanenbaum (2017). *Distributed Systems. 3rd Edition.* 3. Version 01 (February 2017): CreateSpace Independent Publishing.

Villarreal, P. D., I. Lazarte, J. Roa and O. Chiotti (2010). “A Modeling Approach for Collaborative Business Processes Based on the UP-ColBPIP Language”. In W. van der Aalst, J. Mylopoulos, N. M. Sadeh, M. J. Shaw, C. Szyperski, S. Rinderle-Ma, S. Sadiq and F. Leymann (eds.) *Business Process Management Workshops*, pp. 318–329. Berlin, Heidelberg: Springer Berlin Heidelberg.

Weber, I., X. Xu, R. Riveret, G. Governatori, A. Ponomarev and J. Mendling (2016). “Untrusted Business Process Monitoring and Execution Using Blockchain”. In M. La Rosa, P. Loos and O. Pastor (eds.) *Business Process Management: 14th International Conference, BPM 2016, Rio de Janeiro, Brazil, September 18-22, 2016. Proceedings*, pp. 329–347. Cham: Springer International Publishing. URL: https://doi.org/10.1007/978-3-319-45348-4_19 (visited on 11/12/2017).