Process Model-based Access Control Policies for Cross-Organizational Data Sharing

Liam Tirpitz
RWTH Aachen University
Aachen, Germany
tirpitz@cs.rwth-aachen.de

Leon Gentges
RWTH Aachen University
Aachen, Germany
leon.gentges@rwth-aachen.de

ABSTRACT

Controlling who can access which data and when is an important part of enabling data sovereignty during data exchange across organizations. Following the FAIR principles, sharing data with additional metadata, such as provenance information or data quality measures, is a requirement for data reuse. However, in business contexts collaborators often consider the danger of revealing trade secrets through oversharing. Since modern, interconnected processes along or across supply chains hinge on efficient data exchange, easy mechanisms to share valuable data without giving up control are required. We argue that the context of the underlying process, in form of process models, can be used to augment data sharing systems. This can be realized by giving access to data based on visually modeled interactions and by selectively revealing process model fragments as part of the shared metadata. Therefore, we present a framework to define access control policies for collaborative sharing in data ecosystems, based on interorganizational process models. With this approach we take the first step towards tightly integrating process and data management across organizational boundaries.

VLDB Workshop Reference Format:
Liam Tirpitz and Leon Gentges.
Process Model-based Access Control Policies for Cross-Organizational Data Sharing. VLDB 2024 Workshop: 13th International Workshop on Quality in Databases (QDB'24).

1 INTRODUCTION

With increasing digitalization in many domains, data generated by diverse processes has become ubiquitous. Consequently, many companies across various industries actively look for opportunities to efficiently use these large amounts of data. Often, this data is viewed only as an internal asset to optimize local processes and product quality [9], while sharing data is seen with skepticism [18].

However, increasingly complex multi-stakeholder settings arise in many fields, which require cross-organizational collaboration, including research [42], medicine [41] and industrial manufacturing [29]. For example, modern manufacturing processes often produce products from components assembled by separate, specialized departments or external suppliers. The resulting supply chains are complex and interdependent, reaching from the production of the first part to a final product. To enable advanced use cases, such as a circular economy reducing resource consumption, transparency, and traceability of parts is crucial, even through the usage and recycling phase of a product life cycle, which may also be required by legislation in the future [40]. Sharing data across internal departments and external business partners can be an enabler to utilize untapped synergies, improve efficiency, and sustainability and reduce cost. To comply with future regulations and enable frictionless collaborations, manufacturing companies need to adapt data-driven workflows and incorporate the selective publication of high quality manufacturing data into their processes [29].

However, the exchange of data between organizations often introduces friction itself and many obstacles remain that impede the widespread and simple implementation of data sharing in industry [9], caused by organizational hurdles in strategic, regulatory, and operational regards and lack of interoperable technologies. The party who shares data (data provider) is often interested in sharing only selective parts of data and providing secure access to specific parties, to protect potential trade secrets. On the flip side, the data consumer wants to receive as much additional information (metadata) as possible to properly assess the origin and context of the received data and assure that data quality is maintained [22]. As a result of this goal conflict, the details of data and metadata exchanged in supply chains are often manually negotiated between individual collaboration partners, leading to inflexible collaborations.

Initiatives and projects, such as the International Data Spaces [27] or the *Internet of Production* [29] try to break the resulting data silos and strive towards dynamic collaboration and data sharing across organizational boundaries, e.g., by establishing cross-organizational data ecosystems [10]. Such solutions for sharing data usually consider generation and exchange of data as independent operations, while data, especially in the manufacturing industry, is connected to a larger context in form of a physical product the raw materials and the overall process that generated the data. To be useful in collaborations, data needs to fulfill certain requirements, as, e.g., stated by the FAIR principles [42]. These principles require data to be described with rich metadata and provenance to be findable and reusable across contexts. In many business domains, e.g., in industrial manufacturing, data is generated by underlying processes. As the processes influence which data is generated and how, they are important sources of provenance information. We argue that process models can be used to link data to their context, providing a way to explore contexts and find data, contributing to findability and reusability. By advancing FAIR data management and automatically collecting process data across organizations, we can enhance data quality [34], through rich metadata and context information.

Proceedings of the VLDB Endowment. ISSN 2150-8097.

Leveraging detailed product information from suppliers, enriched with relevant details about the production process, such as used raw materials, process parameters, and machines, enables independent assessments on the physical product and the data quality. This may, e.g., be crucial to optimally recycle products. In the long term, industry-wide applied transparency could substantially contribute to establish a circular economy [20] and provide the ability to certify claims on a product (origin, processing, used materials) through selective sharing of production context information formalized in process models.

Data is increasingly considered in the context of process models in literature. For example, Pullonen et al. [30] discuss process modeling to help analyzing privacy leakages along business information flows. Beverungen et al. [2] argue that data generated during processes should be tightly integrated with the respective process models to tackle the challenges of highly connected environments.

We argue, that a multidimensional view on cross-organizational collaborations which considers both, the interconnected processes as well as the exchanged data, in an integrated system can improve the positive impact of collaborations on data quality. By increasing automation, we enhance the quality of context information. Through considering data sharing as a first-class citizen in the execution of the respective business processes themselves, sharing of useful data becomes easier and therefore more viable. In turn, the automated collection and cross-organizational sharing of important context information and process-driven data management provides reliable access to rich metadata which enables organizations to better assess the data quality in collaborative environments. In the other direction, the value of data in process models is increased by connecting them to actionable data ecosystems.

In this paper, we specifically contribute towards sharing of meaningful data, by increasing the integration between processes and the data they generate. By proposing access control policies for data sharing based on the process context of shared resources described through process models, we lower the barrier to share context information, increasing the ability of collaborators to assess the origin and quality of third-party data.

Contributions. Overall, with this paper, we make the following contributions:

- We describe a framework for access control policies which enables data resources to be shared to selected parties based on collaborations modeled in BPMN process models.
- We show an abstraction algorithm to obfuscate the details of a process model shared with third parties, to selectively reveal relevant metadata and context information from a process model, without revealing trade secrets.
- We present our access framework using open standards and protocols as our initial steps towards the vision of user-friendly, automated and FAIR exchange workflows for data ecosystems through collaborative, cross-organizational process modeling.

Structure. In the following, we will first discuss related literature, with regard to access control, process models, and data sharing in Section 2. In Section 3, we discuss our visual access control modeling elements and formalize our underlying access policies in Section 4. Selectively giving access to context information is part of our access policy framework, therefore we present an hierarchical approach to abstract away sensitive details from process models in Section 5. We present our prototypical implementation of process model-based access control in Section 6 and evaluate it in Section 7. We discuss potential future work in Section 8 and conclude in Section 9.

2 RELATED WORK

Our work combines process models with selective data sharing via access control. While literature has mostly addressed these issues individually, there has been work done that combines relevant aspects, such as process model extensions to improve data handling.

2.1 Data in Process Models

While some process modeling languages only consider the *control flow* of a process (i.e., the order of activities), more powerful languages provide basic modeling concepts for data and their exchange across organizations. For example, the Business Process Model and Notation (BPMN) language [26] natively differentiates different types of persistent and non-persistent data objects and collections, which can be connected to activities and control flow elements through data associations. In BPMN, such modeling elements are user-friendly, but do not provide the semantics to formally specify the data interactions needed to automate data handling. While a centralized process execution engine can define its own semantics, distributed, collaborative sharing frameworks need to define meaning for actionable models.

Hund et al. [14] implemented a web-based open source data sharing framework for executing BPMN models across organizations in the healthcare domain. Their implementation focuses on event-based modeling in BPMN and uses a domain-specific data format and underlying infrastructure, which cannot be generalized for other domains. In 2021, Yoshiuchi et al. [43] proposed a data sharing system for manufacturing environments leveraging process models. Their approach concatenates selectively abstracted process model parts from each organization to a shared view which other organizations can run information discovery on and request data access from other participants. While their models do not explicitly handle data access policies and use custom process models, it provides different levels of model abstractions.

2.2 Process Model Abstractions

Similarly to the data resources themselves, their context (in the form of process information) can be confidential, especially to external collaborators. In such cases, the shared context information should not contain all the details of the internal process model. Instead, BPMN process models can be abstracted, such that they only contain the information that is relevant for the partner organization. Smirnov et al. [36] provides a method to generate non-hierachical abstractions through a set of transformations applied based on process semantics and manually selected activities to be hidden. The approach by Ramos Merino et al. [31] recognizes redundant patterns in models and provides appropriate simplifications. Tsagkani and Tsalgatidou [38] focused in their rule-based abstraction approach especially on increasing the comprehensibility of complex BPMN orchestrations. While these approaches provide abstraction

logic for process models, they focus on optimization and improving readability and do not consider the need to hide process details from interorganizational collaborators. Liu et al. [23] highlight the importance of interorganizational processes and mine cooperative process models for collaborating organizations by building common, abstracted process models through a trusted third-party without revealing event logs and process details to collaborators. This approach considers only those activities as shareable, that facilitate communication with external organizations. There is no way to explicitly model access or consider the context of a communicating activity as part of the shareable model.

2.3 Modeling Security in Process Models

Leitner and Rinderle-Ma [21] systematically study different approaches to extended BPMN by security paradigms, such as modeling elements to visualize and enact security concepts, of which access control is one aspect. The analyzed approaches are categorized by their control models, such as role-based and task-based access control. Role-based access control assigns permissions based on user-roles, task-based access control focuses on allowing specific operations. Overall, those modeling elements for process security focus on access control for executing process models, but do not consider the sharing of data and process context. SecBPMN [33] provides a framework with modeling elements to express and enact security policies between actors. However, it does not support interorganizational processes. Kang et al. [19] provide access control to process models, including data access, but do not consider the provenance provided by the process context.

2.4 Provenance-Driven Access Control

Modern access control often falls in the class of Attribute-Based Access Control (ABAC) [13], which is characterized by its ability to consider diverse attributes, including any resource content, system states, contextual information (e.g., user information), with rules expressed through machine-interpretable logical statements. One recent representative is the Access Control Policy (ACP) data model [3], which expresses access policies for resources in RDF. SHACL-ACL [32] uses data shapes to express access control conditions as integrity constraints on RDF graphs.

Managing access control for data based on context information and metadata has been researched in terms of provenance-driven access control, e.g., based on the established PROV data model [24]. By enforcing access control based on data origin and lineage, this approach can efficiently restrict accesses crossing organizational boundaries [1]. Park et al. [28] proposed a family of *Provenance-Based Access Control* (PBAC) models for arbitrary target resources, by defining dependency patterns on the provenance graph via regular expressions. Sun et al. [37] extended this approach and formalized their provenance dependency pattern language as the so called Typed Provenance Model, which defines a start object node and a subsequent provenance pattern to determine access to the target at the end of the path. Danger et al. [7] developed an XML-based access control language for provenance data, and formalized it for the Open Provenance Model (OPM), an earlier representation preceding W3C's PROV. Queries to the provenance graph are responded to with abstracted views that hide nodes or graph fragments as dictated by the provenance access control. Their aim was to ensure that a returned provenance graph view remains a valid provenance graph and does not become semantically invalidated by nodes and edges. These approaches are similar to process-based access control, since they consider the context of data and how it was created. While a relationship between process models and provenance information exists [5], provenance is usually retrospective, while processes are prospective. Therefore, process model-based access policies enable visual and actionable models of collaborations before concrete data elements are generated. To incorporate process context into data ecosystems, access control must consider conventional access policies combined with context-based access via prospective and retrospective provenance in a single model, which currently does not exist.

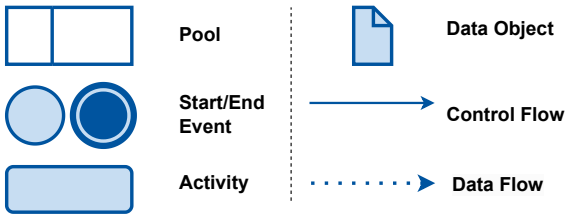


Figure 1: An overview of the used BPMN modeling elements

3 MODELING INTERORGANIZATIONAL PROCESSES WITH DATA EXCHANGE

To apply access control to resources based on process models, we first need to define modeling elements and interaction patterns that we use to express who shares which data with whom and how. These patterns need to be realized through an underlying data ecosystem to translate to functional interactions, which we discuss further in Section 6.

Loose Coupling. Interorganizational processes can be modeled in different ways, ranging from executing a centralized workflow on external resources to loosely coupling independent processes through messages [6, 39]. We use persistently identified, versioned data resources to loosely couple organizations. Interaction between organizations are therefore always read or write operations to data resources and we intentionally omit the typical message flow model syntax used in BPMN. Compared to messaging, resource-based interactions entail a more general notion of loose coupling. The read operation to a resource may (or may not) refer to an already existing resource or revision, without the need to wait for a specific message. This data-focused approach to collaborative interactions still allows message-like communication patterns, for example through a publish/subscribe pattern, such that a message would be realized through triggering a subscription on a data resource or a topic on a message broker. This pattern is chosen, due to its flexibility and suitability for alignment with data sharing ecosystems and we express it through visual modeling elements in BPMN.

Modeling Elements. In BPMN, *pools* (see Figure 1) can be used to encapsulate processes belonging to independent participants. We use

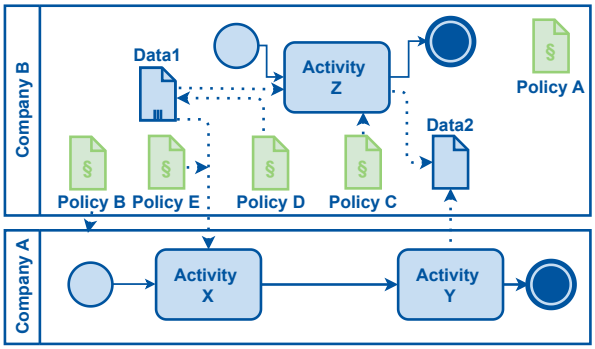


Figure 2: We define multiple modeling patterns to express data exchange and different access policy targets in a process models. Organizations such as Company A are visualized as *pools* embedding their respective process models.

pools to denote *authorities*, which represent organizations responsible for the contained process and data, as illustrated in Figure 2. Authorities need to be uniquely identified by a persistent identifier. *Data Objects* are used to denote data resources (Data2 in our example) or collections of data resources (denoted following the BPMN specification by three vertical lines, see Data1), such as directories. Each data object needs to be uniquely identified within its authority, such that the combined identifier provides a globally unique reference to each resource. Further, data resources may be versioned, so version identifiers can be added, such that access policies target either a specific revision, the latest version, a range of versions or the whole resource, including all revisions. Data Objects can be associated with *activities*, which express the control flow of processes and can read, write, and transform data resources. We define **policy objects** as specific types of data objects, which represent access policies to be applied in the model. To connect these modeling objects, we define patterns on how to express the data exchange associations.

Modeling Patterns. Cross-Organizational data associations can either be reading (Activity X) or writing (Activity Y). The data owner can associate data access policies in the process model. If a policy is modeled inside the pool, but not associated with another modeling element, the policy is applied to all data shared by this authority (Policy A). A policy can be associated with another pool (Policy B). In this case, the policy is applied to all data shared with this authority. If a policy is associated with a data object (Policy D), it applies to all shares of this data, independent of the collaborating authority. Policies associated with data associations (Policy E) only apply to this specific share. Finally, policies associated with activities (Policy C) can be applied to data used by or generated from this specific activity.

These policies and their targets in the process model need to be able to express which data can be read by whom and which context information of a data object should be revealed as part of its metadata. To answer these challenges, we next define our access policy framework.

4 PROCESS MODEL-BASED ACCESS CONTROL

As discussed previously, we define access control policies in different contexts in our process models. To realize these different contexts, we develop a suitable access control framework, by first defining core concepts of our data model. To consider access to resources and their context, we need a combined model considering ABAC, PBAC, and process model context, including selective sharing and hiding of process context information. Next, we present our access control framework which we build in alignment with ACP [3], before discussing its implementation in Section 6.

Access Modes. We define a linear sequence of access modes. The resulting order defines the *restrictiveness* of each mode compared to another, to resolve conflicts later on. We introduce a minimal access mode that allows nothing, access modes allowing to know about the existence of resources (allows resources to be listed in shared data catalogs or visible in process models), and access modes to read or write resource revisions. We assume that each access mode includes the less restrictive modes. For example, the permission to write to a resource also allows reading it.

Target Groups. Instead of defining a an Access Control (AC) for each organization that shall have access to a resource, organizations can be grouped. In that case, a pool in our diagram is not associated with an organization, but a group, which can be indicated through the multiple participant marker (three vertical lines) in BPMN. This allows us to express 1:n manufacturer-to-customer relationships where the manufacturer regularly gets a new customer who may access the same base information as any other customer. Furthermore, we define a group that represents the public. An AC becomes effective if and only if the requesting organization is member of the specified group and if all defined conditions are fulfilled. Then the specified access mode is granted to the target resource.

Conditions for Resource Access. Our model must be able to check if attributes of a target resource satisfy conditions stated in the access policy. In ACP [3] multiple conditions are checked by grouping through `acp:ifAll`, `acp:ifAny` and `acp:ifNone`. An AC can only be effective if the conditions $\Phi_{all}, \Phi_{any}, \Phi_{none}$ are fulfilled in the following way:

$$\begin{aligned}(|\Phi_{all} \cup \Phi_{any}| > 0) \;&\wedge\; (\forall \varphi \in \Phi_{all} : \varphi) \\ &\wedge\; (\Phi_{any} \neq \emptyset \Rightarrow \exists \varphi \in \Phi_{any} : \varphi) \\ &\wedge\; (\forall \varphi \in \Phi_{none} : \neg\varphi)\end{aligned}$$

While empty formula sets evaluate to true, to avoid granting access to resource via empty formulas, we require Φ_{all} or Φ_{any} to have a rule defined that needs to be fulfilled. If Φ_{all} is not empty but Φ_{any} contains no formula, by default $\exists \varphi \in \Phi_{any}$ would always evaluate to false so that the concatenated formula sets would be unfulfilled as well. Thus we make it a condition that empty Φ_{any} can be fulfilled as well. We build our approach on top of this model.

Revisions and Provenance. One dimension to limit access to data is the access to resource revisions. We enable policies to target specific revisions and ranges of revisions, e.g., based on time. For

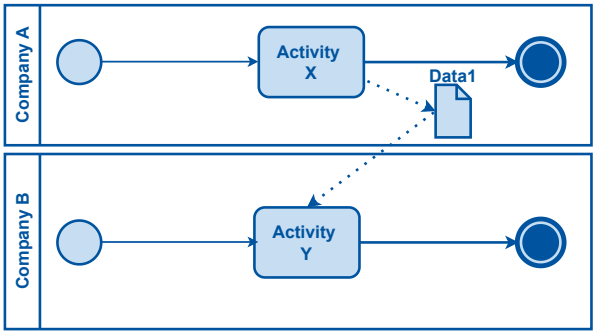


Figure 3: Organization A makes data resource Data1 available to Organization B through actively modeling a data association in a collaborative process model.

example, this can be useful if a resource represents data associated with a product manufactured for a customer on a specific day and all related data should be shared. Further, data may only be shared after a grace period if it becomes less sensitive over time and recycling may become important for a product at the end of its life cycle. In ABAC we can involve environmental values, such as the current time, into condition evaluation. To grant access to a resource effective from, e.g., 5 years after production, one would compare the current time against a resource property that expresses the elapsed time since its generation or a resource attribute stating a confidentiality time limit.

Unfortunately, ACP can only test the existence of an attribute but not compare any values. Therefore, we align the necessary concepts with the Open Digital Rights Language (ODRL) [15]. With its class `odrl:Constraint` an operator (equals, not equals, lower than, greater then, in etc.) and two operands can be represented and evaluated. Either these operands are static strings, dates, or numeric values, or they represent functions that refer to resource values. However, reading a custom value is not natively supported by ODRL, which is required for our model to express access based on provenance relations.

Thus, we introduce a custom `Constraint` class that represents a second-order comparison operator and its operands. The operands can either be static values, a dynamic reference to a resource property (e.g. revision ID), or a custom resource attribute, containing, for example provenance relations, or a reference to an environmental attribute such as the current time. These constraints can be used as regular condition for ACs and concatenated as shown in the formula above. A constraint can be formalized as $\varphi \equiv o_l \circ o_r$ with operand $\circ \in \{=, \neq, <, \leq, >, \geq\}$ and left and right operands o_l, o_r being functions mapping either to static or dynamic numbers, date or string values or sets of the previous types. We can also apply the set operators $\circ \in \{\in, \notin\}$.

BPMN relations. We derive access policies from resource relations depicted and represented in BPMNs as illustrated in Figure 3. To collaborate, organizations can create common process models. These collaborations can be considered as contract, that applies to the resources that were created in context of the respective process instance. As shown in Figure 3, by instantiating the process in the `Company A` pool, A agrees to share the resource instance of Data1 with organization B based on the fact that there is a data association drawn from that resource to an arbitrary task within the organization pool of B. To harden the constraint and prohibit external manipulation, one could additionally demand that the collaboration process model originates from the data provider, in this case A.

Beyond direct modeled sharing relations, we also enable *inheritance*, to derive access and visibility of the surrounding process context. Depending on the target of a policy, different semantics are possible. For example, policies of type A (cf. Figure 2) do not reference specific resources. Instead, all resources contained in a pool inherit the policy.

For policies of type C and type D, targeting specific resources or activities, inheritance can also be applied along the control flow of the process model. This is specifically useful to selectively reveal the provenance of a shared resource, based on the process context. For example, if we share a resource, we may also want to share the details of the generating activity and the resources that were used by the generating activity. By defining the reach of a policy along control flow relations, we can define which part of a process model and the connected data resources should be fully or partially affected by a policy.

We define an *inheritance direction* to be *forward*, *backward*, or *both*. A policy is inherited forwards, from an origin to a target, if and only if there is a path of control- and dataflow relations connecting the origin with the target. Analogously, a backward inheritance is applied if the target is connected to the origin. Note that this definition also includes paths along cycles. We define a *path length* to limit the scope of shared context information. Specifically, a policy is only inherited, if the path between the origin and target is no longer that the given path length. Further, a policy inheritance can be set to inherit the full access policy, or only influence the revelation of the process context, which we discuss in Section 5.

Formally, we extend our access control resource by a set H that stores zero or more reference functions that describe the environment of a BPMN modeling element. Each refers to a set of matched resources. To determine the access permission mode of the target resource, the granted permissions for the related resources are determined. The minimum of these permissions is then taken into account as permission for the target resource. If there are policies for the specific target resource defined and the permission granted by the resource-specific policies are more restrictive than the inherited ones, that more restrictive permission overwrites the inherited ones: Let $A_o(f)$ return the the resource-specific, granted access mode for resource (revision) f to organization o.

$$A'_o(f) = max\{A_o(f), min\{A_o(h(f')) \mid h \in H\}\}$$

then returns the applied access permissions.

Resolving Conflicting Rules. Since policies can target different modeling elements, multiple policies can apply for the same interaction, potentially leading to conflicts. To resolve this, we prioritize the more specific policy type, e.g., a policy that is valid for all data objects in a pool is less specific then a policy assigned to a specific data object. We denote $x <_s y$ to indicate that x is more specific than y

and $x =_s y$ to denote that x and y are equally specific. For the policy types defined in Section 3, we define $E <_s D <_s C <_s B <_s A$.

Note that the *specificity* of a policy determines *if* a certain element should be affected, while the *restrictiveness* is used to determine which policy should be applied if multiple policies affect the same element. For example, different models may reference the same data objects, which can lead to multiple, equally specific policies. In that case, our access control model is evaluated the following way: A target resource is prohibited to be accessed unless a an AC is found whose conditions are all fulfilled ("deny unless permit" [25]). This cautious evaluation approach minimizes the risk of unintentionally revealing sensitive information by ensuring that no prohibition rules are overlooked during the definition process. By design, also the combination of conditions/constraints within an AC is unambiguous.

However it is possible to define cyclic inheritance dependencies. For example, evaluating an AC A that depends on an AC B that, again, depends on AC A would not terminate. Users need to consider that when defining ACs. However, we will not go deeper into that issue since it does not constitute a security threat where resources might unintentionally be exposed.

5 SELECTIVE MODEL REVEALING

While data providers are interested in providing their process models for smooth collaboration, sensitive details should remain hidden and data consumers shall only have access to abstracted views.

The data owner can create multiple different views of a process, each dedicated to specific groups of collaborators and select, which elements (*abstraction objects* [35]) may be included in a view, and thus be exposed. This can be done through explicitly choosing exposed modeling elements. Additionally, we chose to automatically expose the modeling elements in a certain range around a shared resource as part of the provenance metadata, if defined in a policy. The opposite approach of *selective hiding* where the owner manually marks which aspects of the model should remain hidden in the view carries a greater risk of implicitly not hiding sensitive information. Since we focus on data sharing, we develop a BPMN abstraction that considers both, control and data flows.

Hierarchical Abstraction. We explore a scenario where a less sensitive data resource is embedded within a sensitive process fragment. The shared view should not contain this sensitive process fragment and the data it typically generates. However, we must be able to reveal the less sensitive data resource itself, irrespective of the sensitive process fragment in the view. While fine-grained control over data resources is required, the degree of detail in abstracting the control flow elements is less relevant for data-centric collaboration.

For our abstraction approach, we exploit the hierarchical structure that sub-processes in BPMN diagrams offer. In line with common organizational structures, we organize processes hierarchically [17]. We argue that on a large scale, our sub-process based abstraction method is feasibly transferable to real-world applications. However, the price of our method is that process engineers need to actively pay attention to model that hierarchical depth in the process models since flat process models cannot be abstracted at a very granular level.

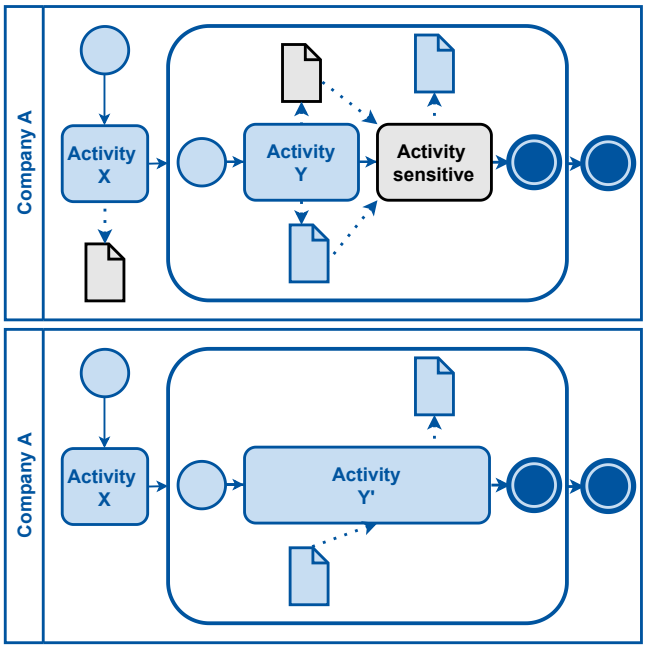


Figure 4: Example of our hierarchical BPMN abstraction approach: Original process model of *Company A* (top) and resulting view (bottom).

We define a *process layer* as the elements of a process or a sub-process between a start and one or multiple end events without stepping into sub-processes or crossing organizational borders (pools). Note that out of the set of considered elements, crossing organizational borders is only possible for data associations. On each process and sub-process layer, after execution of our abstraction the following two rules must hold true:

- The data flow elements (data object and data associations) of the current process layer will be included in the abstracted view if and only if they are marked as visible.
- The control flow elements (tasks, sub-processes, gates, sequence flows) of the current layer will be included in the abstracted view if and only if all tasks and sub-processes of the current layer are marked as visible. All invisible control flow elements in that layer are abstracted into a single task.

Whether a process element is marked visible or not, is derived from the access permissions assigned to its respective resource instance. Our access control model, for example, assigns an access mode to the access policies, determining whether another organization may be aware of the existence of a resource.

To enable the process of model abstraction, we define the following operations on a BPMN model: deletion of elements, re-associating data objects, and creating simple processes consisting only of a connected start event, a new dummy task, and an end event. This allows well defined and compact transformation, which could be expanded in the future.

In the following we formalize an algorithm that abstracts a BPMN model based on the previously formulated rules. The algorithm outputs a valid abstraction of the original BPMN. When observed

externally, without insight into internal operations, BPMN sub-processes function similarly to conventional tasks. To stress that behavior, in the following we name sub-processes as *group tasks*.

We formalize a given process model m as an element in the set of all BPMN process models PM as $m = (P, T, G, D, C, F, S, E, \sigma, \tau) \in \mathsf{PM}$, where:

- P is a non-empty, finite set of pools.
- T is a non-empty, finite set of tasks
- G is a non-empty, finite set of gateways (decision points, splitting or merging control flows, e.g., AND gates).
- D is a finite set of data objects.
- C is a finite set of directed control flow edges (relations connecting elements in S, T, G, E within a pool).
- F is a finite set of directed data flow edges (a relation associating elements in T, D with each other).
- S is a non-empty, finite set of start events.
- E is a non-empty, finite set of end events.
- $\sigma : P \cup T \rightarrow \mathcal{P}(T \cup G \cup D)$ maps from conventional BPMN tasks to $\emptyset$ and from process layer wrappers (pools in P and group tasks $t_g \in T_G \subseteq T$) to the set of its contained BPMN elements.
- $\tau : P \cup T_G \rightarrow S$ assigns a start event to pools and group tasks.

With $\mathtt{include}_v : T \cup D \rightarrow \{true, false\}$, we define, what BPMN elements from $T \cup D$ shall be included in view v of $m \in \mathsf{PM}$. In the example input BPMN in Figure 4, data objects and tasks in the *abstraction set* [35] $\omega = \{e \in D \cup T \mid \mathtt{include}_v(e) = false\}$ are colored gray and thus are not included in view v.

The resulting view (bottom) has run through the following reductions: Since the sub-process (second level) contains at least one *gray* sensitive control flow element, the whole control flow in that sub-process is replaced by a dummy task. Since all control flow of the process on first level (pool content) are in-sensitive, its control flow elements are not abstracted.

To abstract v from the original process collaboration m, we apply the following Algorithm 1 with $\mathtt{include}_v$ as input.

The presented Algorithm 1 starts with the original process model view as input which it reduces (line 1) afterwards. First it iterates all pools in the BPMN collaboration (line 2) to recurse into their nested process levels and reduce the control flow on each level depending on the elements' visibility (line 3). Afterwards, all control flows in the process collaboration are still valid and contain only elements that are marked as visible with $\mathtt{include}_v$. Finally (line 5), isolated from the control flow abstraction, all data objects that are not marked as visible (line 30) and their connected data associations (line 31 and 32) are purged from the process model view v.

With the process levels (group tasks) considered as nodes of the process hierarchy search space which spans starting at each pool, the control flow abstraction executes a *depth first search* on the hierarchy levels (lines 11 and 14). For a current process level, the member tasks T are determined (line 8) and filtered for group tasks T_G (line 9). Subsequently to the recursion into the subordinate process layers, the abstraction condition whether all control flow elements in the current process layer are visible is checked (line 10). If it is not fulfilled, the control flow contained in the current process layer gets abstracted to/*replaced by* a new "dummy" process consisting of a start event, a task that represents the abstracted tasks of that process layer, and an end event (lines 18 - 22). Data flows are handled independently from control flows. Thus, all data objects that previously were associated with an abstracted (hidden) task, get re-associated/-connected to newly created "dummy" task that replaces the latter (lines 24 - 27).

We have identified another algorithm variant where the creation of such a "dummy" process is skipped. Instead, the abstracted elements are represented by the wrapping group task and data objects that used to be connected to the abstracted tasks are re-associated with the group task.

To simplify visualization and avoid overlapping elements, we aim to solely remove or overwrite existing BPMN elements without

Algorithm 1 Hierarchical Process Model Abstraction

Require: $m \in \mathsf{PM}, \mathtt{include}_v \in T \rightarrow S$
Ensure: $v = (P, T, G, D, C, F, S, E, \sigma, \tau) \in \mathsf{PM}$

1: $v \leftarrow m$
2: **for all** $p \in P$ **do**
3: HierarchicallyAbstractCF(p)
4: **end for**
5: PurgeExcludedDataObjects
6: **return** v

7: **procedure** HierarchicallyAbstractCF($l \in P \cup T_G$)
8: $T' \leftarrow \sigma(l) \cap T$
9: $T'_g \leftarrow \{t' \in T' \mid \sigma(t') \neq \emptyset\}$
10: **for all** $t'_g \in T'_g$ **do**
11: $v \leftarrow$ HierarchicallyAbstractCF(t'_g)
12: **end for**
13: **if** $\exists t' \in T' : \mathtt{include}_v(t') = false$ **then**
14: AbstractCurrentCFLevel(l, T')
15: **end if**
16: **end procedure**

17: **procedure** AbstractCurrentCFLevel($l \in P \cup T_G, T' \subseteq T$)
18: $G \leftarrow \{g \in G \mid gCt' \vee t'Cg \Rightarrow t' \notin T'\}$
19: $E \leftarrow \{e \in E \mid tCe \Rightarrow t \notin T'\} \cup \{e_d\}$
20: $T \leftarrow (T \setminus T') \cup \{t_d\}$
21: $s \leftarrow \tau(l)$
22: $C \leftarrow \{(n_1, n_2) \in C \mid n_1 \notin T' \wedge n_2 \notin T'\} \cup \{(s, t_d), (t_d, e_d)\}$
23:
24: $F'_{in} \leftarrow \{(d, t') \in F \mid t' \in T' \vee \exists t_g \in T' : t' \in \sigma(t_g) \cap T\}$
25: $F'_{out} \leftarrow \{(t', d) \in F \mid t' \in T' \vee \exists t_g \in T' : t' \in \sigma(t_g) \cap T\}$
26: $F \leftarrow F \setminus (F'_{in} \cup F'_{out})$
27: $F \leftarrow F \cup \{(t_d, f'_{in}) \mid f'_{in} \in F'_{in}\} \cup \{(f'_{out}, t_d) \mid f'_{out} \in F'_{out}\}$
28: **end procedure**

29: **procedure** PurgeExcludedDataObjects
30: $D' \leftarrow \{d \in D \mid \mathtt{include}_v(d) = false\}$
31: $F'_{in} \leftarrow \{(d', x) \in D' \times T \mid d'Fx\}$
32: $F'_{out} \leftarrow \{(x, d') \in T \times D' \mid xFd'\}$
33: $F \leftarrow F \setminus (F'_{in} \cup F'_{out})$
34: $D \leftarrow D \setminus D'$
35: **end procedure**

introducing new ones. Given that a top-level process typically only has a pool as its enclosing element, which in contrast to tasks cannot be the target of data associations, the pool is abstracted as a dummy process. The same applies to incoming data associations from other pools. This approach offers the advantage of retaining meaningful data object associations within the diagram, as data objects cannot be associated with black-box pools.

Based on this BPMN abstraction method we can in practice filter out sensitive process elements from shared process model views based on the regular access control of the resources generated by the process model instances.

Sharing Context-dependent Abstractions. As discussed in Section 4, we enable policies to define a context *scope*, which can automatically inform which parts of a process model and the associated data resources should be shared as part of a given data exchange. This can then be used to identify insensitive resources as the input for our hierarchical abstraction algorithm.

6 DATA ECOSYSTEM REALIZATION

To use the previously discussed access policies and automate interorganizational data exchange, we map the process model concepts to data exchange primitives in executable software. Specifically, we base our realization on the FactStack by Gleim et al. [12]. The FactStack provides exchange mechanism for data resources based on semantic and standardized web technologies. In the underlying, resource-based FactDAG data model [11], each revision of a versioned resources is called a *Fact*. *Authorities* model organizations, which are responsible for their data. Each Fact is persistently identified by a triple consisting of an *authorityID*, a *resourceID* and a *revisionID*. In the FactStack implementation, FactIDs are realized as URIs, which are resolvable through HTTP(S). Each sovereign authority provides managed access to their data via a Linked Data Platform (LDP) server. The FactDAG also provides a provenance model aligned with PROV [24], which links *process steps* and their executions to Facts across organizational boundaries. By combining persistent identifiers, provenance, and web-based data access the resulting FactDAG provides sovereign data exchange and tracing of data origin, e.g., along supply chains.

Identifiers. To implement process-model based access control on top of the FactStack, we need to correctly associate data objects in the process model with actual data stored in the FactDAG. We do so by defining custom BPMN attributes, which assign FactIDs to modeling objects in BPMN. As discussed in Section 3, we assign authorityIDs to pools in BPMN, expressing ownership of a certain organization over processes and data contained in the pool. The FactDAG represents a retrospective model of created data and executed processes, while process models represent prospective *templates* on how data should be generated. Specifically, process models may be instantiated and executed multiple times. If a data object is the output of an activity, each execution may lead to a separate data object. Therefore, data objects in BPMN do not relate to individual data points, but one data point *per process instance*, i.e., for each execution of the process, a different data element may be referenced. Therefore, the abstract data object in BPMN represents a collection of concrete, individual elements of data. Subsequently, we align BPMN data objects with resources in the FactDAG, since resources can be interpreted as collections of Facts, i.e., each revision to a resource is a Fact, just like each instance of a process model references a new element of data. Consequently, resourceIDs always need to be directly associated with a data object in BPMN.

The revisionID is a property of a single Fact. If a data object has only an incoming edge from a task in the BPMN model, the revisionID is assigned during the execution of the activity pointing to the data object. If a data object has an outgoing edge, different points of data could be referenced, because resources are classes of Facts and not specific Facts. By associating specific revisionIDs to a modeled data object, we can reference either a specific revision of a resource, the whole resource, the most recent revision or even a range of revisions.

6.1 Expressing Access Control Policies

Since policies are also data objects in BPMN, we model them as facts in the FactDAG model with a FactID. By referencing the same FactID, we can reuse policies (or facts in general) in different process models. We express our access control policy through an ontology.

6.1.1 Group Resources. To represent groups or organizations, we created a specialization of FactDAG entities named *GroupResource* which is represented in the FactStack.

A group can express containment through `ac:hasMember` attributes whereas the latter has organization Universal Resource Identifierss (URIs) as range. If group information is requested for membership of a group, it will always return true if the group is furthermore assigned the type `ac:PublicGroup`. With that simple, yet extendable definition of groups we can refer to them from Access Control Resources (ACRs) that we will present in the following. We enable organizations to define their own private set of organization groups. In general, we recommend to always refer to the latest revision of a group instead of a specific revision. That way groups can be altered without having to update all resources referring to them. However, depending on the use case, specific revision can be referenced to *pin* a specific group composition for a certain context.

6.1.2 Access Control Resources. ACRs are specializations of resources and are stored in the FactStack, containing the actual policies. Aside of group resources that are intended to be reused by multiple access control policies, the access policy for all revisions of a resource are stored in one ACRs to minimize the number of required resource requests from the server. Only the most recent revision of such an ACR is valid and overriding an ACR (or revisioning it) invalidates the previous revision.

With the `ac:inheritAccessPermissionsFrom` attribute, dependencies can be expressed, pointing to other related target resources (`ResourceReference`, that need to be accessible for the target resource to be accessible as well. In the following we will discuss implementation considerations of the access control model in detail.

Access Control. Access control policies are in our model represented by the AccessControl class. An ACR can have multiple AccessControls that serve as "Authorization" if they are effective. The targetGroup field contains an identifier, pointing to a group resource, which the policy should be applied to. A policy is effective when all conditions are fulfilled by the context: The targetGroup matches

the requesting entity. Furthermore, the previously discussed flat condition combiners `ifAll`, `ifAny` and `ifNone` must be fulfilled. To evaluate complex conditions, such as the targetGroup we need to asynchronously request further resources during evaluation time. Eventually, the evaluation method returns whether AccessControl is effective. If it is effective, the accessMode of the policy is applicable to the target resource for the requesting organization.

Access Modes. To express access modes that an access policy can assign to a resource for a certain group of users, we introduced the following modes, each assigned a numeric weight, such that the *more restrictive* mode has the higher weight:

(1) `ac:Nothing`
(2) `ac:DiscoverResource`
(3) `ac:DiscoverResourceRevision`
(4) `ac:ReadRDF`
(5) `ac:ReadBinary`
(6) `ac:Write`
(7) `ac:Delete`

To check if an `AccessControl` with access mode x allows an operation on the target resource that requires, e.g., access mode *ac:DiscoverResourceRevision*, we compare the weight of x $w(x)$ to the weight of *ac:DiscoverResourceRevision*. As we pursue the strategy of *deny until permit* in our data sharing approach, the default access mode is always `Nothing`. On the other hand, if there are multiple AccessControls that are effective, the access mode of the most restrictive one is granted to the target resource.

Conditions. To express conditions, in our implementation we rely on the `Constraint` class, similar to `odrl:Constraint` [16]. It defines two operands that are either literals such as strings, numbers or dates or the requester authorityID, or placeholders for the current date or resource references that need to be evaluated to literals. Since resource references behave like matchers that return all references they could find, we have constructed `Constraint` in a robust way that it expects all its operands to evaluate to lists of literals.

It offers primitive operators such as `ac:equals`, `ac:greaterThan`, `ac:lessThan`, etc. but also set operators `ac:in` and `ac:notIn`. If the operator is set to a primitive operator such as `ac:lessThan`, the constraint is fulfilled if and only if all left operand entries L combined with all right operand entries R fulfill the operator ($\forall l \in L \forall r \in R : \text{eval}(l) \circ \text{eval}(r)$ with primitive operator $\circ$). If the operator is set to a set operator such as `ac:in`, the constraint is fulfilled if and only if any left operand entry combined with any right operand entry fulfills the operator ($\forall l \in L \exists r \in R : \text{eval}(l) \circ \text{eval}(r)$ with set operator $\circ$). However, if any operator evaluates to empty, then the `Constraint` is never fulfilled.

Access. Through this mapping to the FactDAG model, we relate process model objects to resolvable identifiers and therefore resources (data resources, policies and the process models themselves) that can be accesses via HTTP(S). Similarly our approach could also be extended with mappings to other data ecosystem frameworks. While this requires the modification of process models beyond the current standard, it also enables semantically well defined data sharing operations based on open standards through process models across systems, infrastructures and organizations. In case of the process models, each organization can store the pool associated with the own organization. By modeling third party organizations in the own model, data access (and model sharing) requests could be automatically triggered. This connection between process models and data exchange can already improve the interorganizational data sharing workflow. In the future, the degree of automation could be further increased by also using a process execution engine.

7 EVALUATION

With our implementation we demonstrated the feasiblility of process model driven data exchange on top of data ecosystem infrastructure. To evaluate how well our approach scales with the current implementation, we test our system with increasingly large process model contexts. This evaluation is based on a straightforward, non-optimized proof of concept implementation, through which we can discover targets for future performance optimizations.

Experimental Setup. We set up our experiments as follows. First, we generate collaborative BPMN process models with different process sizes and populate data resources on multiple FactStack instances. Afterwards, we create various access control policies on elements in these processes. From that point on, the system state is not changed anymore so that each test run takes place under equal system conditions. To measure the performance, the resources, that we previously assigned access control policies to, are requested by multiple organizations and the time needed to execute these evaluation requests is taken for each request individually. In our case, we decided to base our benchmark on three different organizations. Finally, the results are plotted as execution time against instance size. We executed this benchmark with the FactStack server as well as the client running on one Linux machine.

Dataset. To control the scale of our processes, we generated synthetic models. Due to the structure of our policy evaluation, which is based on repeated HTTP requests, we identified large models a potential weak point of our current implementation and focused our evaluation on scaling the dimension of process control path.

Starting from an empty process model, we created a BPMN generator that allows to extend that empty process model linearly elements. As a result, we get BPMN models representing the internal XML structure while neglecting any graphic visualization that we do not require four our benchmark. In Figure 5 we show a schematic view of our test instances. Thereby process task chain for organization A has varying length and determines the instance size n. In the following, organization A is named chain owner, organization B customer (partial access allowed and C serves as uninvolved organization whose access request will always be denied.

Results. The results of our performance evaluation are shown in Figure 6. For three organizations the execution time to answer a request was measured for varying instance sizes. The plots indicate the average response time over 10 executions on each test instance.

Figure 6a shows this graph for the data owner, Organization A. The execution time is constant and thus independent of the instance size. Additionally the execution time is, compared to the diagrams we will see in the following, small. This is due to the fact that our access control always grants data owners access to resources based on their own AuthorityID. Thus we have a constant runtime independent of the size of the process instance.

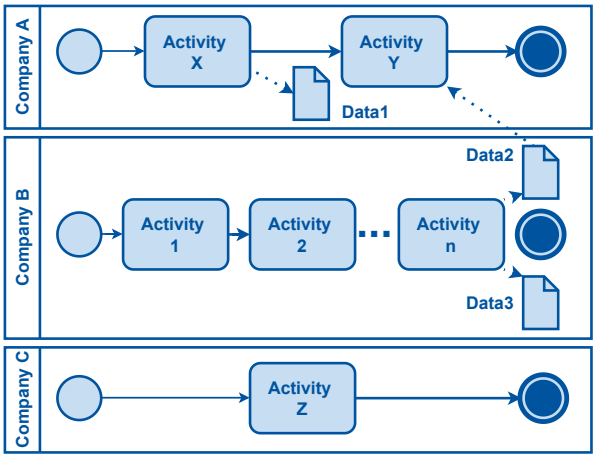


Figure 5: Schematic view of our BPMN collaboration test instance.

Figure 6b shows the execution time for a request from a user (Organization B) that is granted access to the data resource associated with its own process pool but not to others. A linear dependency can be observed for both cases (access denied and access granted). However, in the case of a non accessibility the execution time is lower than in the case of accessibility. The lower execution time can be explained by the prohibition of downloading the element from the server. The accessibility of a resource for an organization is dependent on the existence of a data association going from the target resource to the pool of the respective requesting organization, which is the case for organization B. Since the policy evaluation starts from the requested data resource (Data2) the algorithm traverses the FactDAG in order to request the existence of a respective data association, that fulfills the access condition.

In Figure 6c the execution time for a user request, where a user doesn't have access to any of the requested resources (Organization C), we can observe that the execution time shows again a linear dependency of the instance size.

Impact. As shown in the previous section, the execution time of our policy evaluation reusing BPMN relations decreases with increasing instance size and are high overall. This massively constrains the scalability of of our current, FactStack-based implementation. In the following, we will discuss possibilities to increase this performance.

Our architecture is not optimized to minimize the number of requests made to determine the access permissions of a resource. Thus, time-consuming redundant requests to the same resource are done even, e.g., to the same target resource while evaluating the access control resource. One could tackle that issue if one would consider the state of the FactDAG as frozen for a batch request such as instantly succeeding process model abstractions. A cache map of relevant resources could be kept for a short while. Further, through realizing the retrieval of policies directly through server-side database requests instead of resource retrieval through the LDP interface, the overall performance could be increased substantially.

8 FUTURE WORK

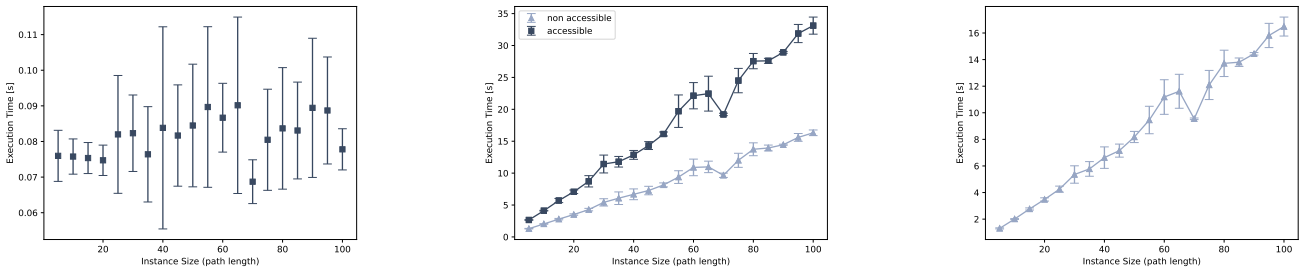
The access control policy framework presented in this paper provides one building block towards our vision of tightly integrated process data and exchange workflows. Next, we intend to focus on improving the user experience and degree of automation.

Data Exploration and Collaboration. While our current implementation provides functional data exchange capabilities by utilizing the FactStack data ecosystem, it lacks a integrated user interface. To lower the barriers to data exchange, users need to be able to easily explore available data and seamlessly integrate it into their local workflows. In the future, we intend to build a distributed web application which enables collaborative modeling of interorganizational workflows and data exchange, e.g., by utilizing existing tools such as bpmn.io [4]. This system could automatically issue data access requests to collaborators, based on a provided collaboration model and provided different, abstracted views to the process models in an interactive environment. By integrating the system with a data catalog, we could automatically publish process (meta-) data and in turn use cataloged data in our process models.

Interoperable Automation and Streaming Data. Currently, our data model assumes data at rest with identifiable data resources, persisted in some kind of data store. While this model, including the versioning of resources, covers many data sharing scenarios, it cannot handle dynamic, near-realtime collaboration scenarios. In the future, we want to extend our model to data stream sources with the goal to dynamically connect data processing pipelines across system-, infrastructure-, and organizational boundaries. Further, our current implementation does not integrate with any sort of execution engine to process data. To connect automated processes and build data pipelines across organizations, we also need to connect our approach to the underlying execution engines and enable data and control to flow interoperably between these independent systems. To support the necessary data interoperability, we want to look at shape-based solutions to model and validate evolving data generated during processes, such as PALADIN [8] and align our interorganizational, process model-based approach with data shape languages for access control, such as SHACL-ACL [32].

9 CONCLUSION

The increasing drive towards integration of processes across organizational boundaries and the growing amount of data, e.g., in the manufacturing industry [29], leads to the need for frictionless, integrated exchange of interoperable data. Addressing this need, we argue that process models can play a crucial role in interorganizational cooperation in many industries, such as research, health or manufacturing. Specifically, cross-organizational processes can be visualized and data exchange can be automated through interactive and cooperative modeling, leading to increased collection, interconnection and sharing of context information, which enables collaborators to assess data quality by tracing its provenance. In this paper, we present process model-based access control policies as a core building block towards a fully integrated process data management system. We present a BPMN modeling extension to visually assign sharing policies in process models and by explicitly



(a) Policy evaluation times for org A. **(b) Policy evaluation times for org B.** **(c) Policy evaluation times for org C.**

Figure 6: Policy evaluation times for all organizations in our test model, over different model sizes.

considering access to context information in form of process models, we enhance the quality of the shared data. Therefore, with our access policy framework, models define the scope of a data sharing operation and are also part of the shared context information, supporting both, automation and data quality.

Overall, our work contributes to a vision of flexible and interoperable cross-organizational collaboration and tight integration of processes and their data, utilizing the context information provided by process models to enhance data exchange.

ACKNOWLEDGMENTS

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC-2023 Internet of Production - 390621612.

REFERENCES

[1] Adam Bates, Ben Mood, Masoud Valafar, and Kevin Butler. 2013. Towards Secure Provenance-Based Access Control in Cloud Environments. In *Proceedings of the Third ACM Conference on Data and Application Security and Privacy (CODASPY '13)*. Association for Computing Machinery, New York, NY, USA, 277–284. https://doi.org/10.1145/2435349.2435389

[2] Daniel Beverungen, Joos C. A. M. Buijs, Jörg Becker, Claudio Di Ciccio, Wil M. P. van der Aalst, Christian Bartelheimer, Jan vom Brocke, Marco Comuzzi, Karsten Kraume, Henrik Leopold, Martin Matzner, Jan Mendling, Nadine Ogonek, Till Post, Manuel Resinas, Kate Revoredo, Adela del-Río-Ortega, Marcello La Rosa, Flávia Maria Santoro, Andreas Solti, Minseok Song, Armin Stein, Matthias Stierle, and Verena Wolf. 2021. Seven Paradoxes of Business Process Management in a Hyper-Connected World. *Business & Information Systems Engineering* 63, 2 (April 2021), 145–156. https://doi.org/10.1007/s12599-020-00646-z

[3] Matthieu Bosquet. 2022. Access Control Policy (ACP). https://solidproject.org/TR/acp

[4] bpmn.io. 2023. Web-based tooling for BPMN, DMN, CMMN, and Forms | bpmn.io. Retrieved August 3, 2024 from https://bpmn.io/

[5] Anila Sahar Butt and Peter Fitch. 2021. A Provenance Model for Control-flow Driven Scientific Workflows. *Data & Knowledge Engineering* 131 (2021), 101877.

[6] Issam Chebbi, Schahram Dustdar, and Samir Tata. 2006. The view-based approach to dynamic inter-organizational workflow cooperation. *Data & Knowledge Engineering* 56, 2 (2006), 139–173. https://doi.org/10.1016/j.datak.2005.03.008

[7] Roxana Danger, Vasa Curcin, Paolo Missier, and Jeremy Bryans. 2015. Access Control and View Generation for Provenance Graphs. *Future Generation Computer Systems* 49 (Aug. 2015), 8–27. https://doi.org/10.1016/j.future.2015.01.014

[8] Antonio Jesus Diaz-Honrubia, Philipp D. Rohde, Emetis Niazmand, Ernestina Menasalvas, and Maria-Esther Vidal. 2024. PALADIN: A process-based constraint language for data validation. *Information Fusion* 112 (2024), 102557. https://doi.org/10.1016/j.inffus.2024.102557

[9] Marcel Fassnacht, Carina Benz, Daniel Heinz, Jannis Leimstoll, and Gerhard Satzger. 2023. Barriers to Data Sharing among Private Sector Organizations. In *56th Hawaii International Conference on System Sciences*, Vol. 56. ScholarSpace, Maui, 3695–3704. https://hdl.handle.net/10125/103084

[10] Sandra Geisler, Maria-Esther Vidal, Cinzia Cappiello, Bernadette Farias Lóscio, Avigdor Gal, Matthias Jarke, Maurizio Lenzerini, Paolo Missier, Boris Otto, Elda Paja, Barbara Pernici, and Jakob Rehof. 2022. Knowledge-Driven Data Ecosystems Toward Data Transparency. *Journal of Data and Information Quality* 14, 1 (March 2022), 1–12. https://doi.org/10.1145/3467022

[11] Lars Gleim, Jan Pennekamp, Martin Liebenberg, Melanie Buchsbaum, Philipp Niemietz, Simon Knape, Alexander Epple, Simon Storms, Daniel Trauth, Thomas Bergs, Christian Brecher, Stefan Decker, Gerhard Lakemeyer, and Klaus Wehrle. 2020. FactDAG: Formalizing Data Interoperability in an Internet of Production. *IEEE Internet of Things Journal* 7, 4 (April 2020), 3243–3253. https://doi.org/10.1109/JIOT.2020.2966402

[12] Lars Gleim, Jan Pennekamp, Liam Tirpitz, Sascha Welten, Florian Brillowski, and Decker Stefan. 2021. *FactStack: Interoperable Data Management and Preservation for the Web and Industry 4.0.* Gesellschaft für Informatik, Bonn. https://dl.gi.de/handle/20.500.12116/35804

[13] Vincent C. Hu, David Ferraiolo, Rick Kuhn, Adam Schnitzer, Kenneth Sandlin, Robert Miller, and Karen Scarfone. 2014. *Guide to Attribute Based Access Control (ABAC) Definition and Considerations.* Technical Report NIST SP 800-162. National Institute of Standards and Technology. NIST SP 800–162 pages. https://doi.org/10.6028/NIST.SP.800-162

[14] Hauke Hund, Reto Wettstein, Christian M Heidt, and Christian Fegeler. 2021. Executing Distributed Healthcare and Research Processes - The HiGHmed Data Sharing Framework. *Studies in health technology and informatics* 278 (May 2021), 126–133. https://doi.org/10.3233/shti210060

[15] Renato Iannella and Serena Villata. 2018. *ODRL Information Model 2.2.* W3C Recommendation. W3C. https://www.w3.org/TR/2018/REC-odrl-model-20180215/.

[16] Renato Iannella and Serena Villata. 2018. ODRL Information Model 2.2. https://www.w3.org/TR/odrl-model/

[17] Girish S. Joglekar, Arun Giridhar, and Gintaras Reklaitis. 2014. A Workflow Modeling System for Capturing Data Provenance. *Computers & Chemical Engineering* 67 (Aug. 2014), 148–158. https://doi.org/10.1016/j.compchemeng.2014.04.006

[18] Ilka Jussen, Frederik Möller, Julia Schweihoff, Anna Gieß, Giulia Giussani, and Boris Otto. 2024. Issues in inter-organizational data sharing: Findings from practice and research challenges. *Data & Knowledge Engineering* 150 (2024), 102280. https://doi.org/10.1016/j.datak.2024.102280

[19] Myong H. Kang, Joon S. Park, and Judith N. Froscher. 2001. Access control mechanisms for inter-organizational workflow. In *Proceedings of the Sixth ACM Symposium on Access Control Models and Technologies* (Chantilly, Virginia, USA) *(SACMAT '01)*. Association for Computing Machinery, New York, NY, USA, 66–74. https://doi.org/10.1145/373256.373266

[20] Hervé Legenvre and Ari-Pekka Hameri. 2023. The Emergence of Data Sharing along Complex Supply Chains. *International Journal of Operations & Production Management* ahead-of-print, ahead-of-print (Jan. 2023), 292–297. https://doi.org/10.1108/IJOPM-11-2022-0729

[21] Maria Leitner and Stefanie Rinderle-Ma. 2014. A systematic review on security in Process-Aware Information Systems – Constitution, challenges, and future directions. *Information and Software Technology* 56, 3 (2014), 273–293. https://doi.org/10.1016/j.infsof.2013.12.004

[22] Maria Linnartz, Soo-Yon Kim, Martin Perau, Tobias Schröer, Sandra Geisler, and Stefan Decker. 2022. Unternehmensübergreifendes Datenqualitätsmanagement-Entwicklung eines Rahmenwerks zur Analyse der Stammdatenqualität in Kunden-Lieferanten-Beziehungen. *Zeitschrift für wirtschaftlichen Fabrikbetrieb* 117, 12 (2022), 851–855. https://doi.org/10.1515/zwf-2022-1167

[23] Cong Liu, Hua Duan, QingTian Zeng, MengChu Zhou, FaMing Lu, and JiuJun Cheng. 2019. Towards Comprehensive Support for Privacy Preservation Cross-Organization Business Process Mining. *IEEE Transactions on Services Computing* 12, 4 (July 2019), 639–653. https://doi.org/10.1109/TSC.2016.2617331

[24] Paolo Missier and Luc Moreau. 2013. *PROV-DM: The PROV Data Model.* W3C Recommendation. W3C. http://www.w3.org/TR/2013/REC-prov-dm-20130430/.

[25] Aya Mohamed, Dagmar Auer, Daniel Hofer, and Josef Küng. 2021. Extended Authorization Policy for Graph-Structured Data. *SN Computer Science* 2, 5 (June 2021), 351. https://doi.org/10.1007/s42979-021-00684-8

[26] OMG. 2011. Business Process Model and Notation (BPMN), Version 2.0. http://www.omg.org/spec/BPMN/2.0

[27] Boris Otto, Sebastian Steinbuß, Andreas Teuscher, and Steffen Lohmann. 2019. IDS Reference Architechture Model, Version 3.0. https://internationaldataspaces.org/wp-content/uploads/IDS-Reference-Architecture-Model-3.0-2019.pdf

[28] Jaehong Park, Dang Nguyen, and Ravi Sandhu. 2012. A Provenance-Based Access Control Model. In *2012 Tenth Annual International Conference on Privacy, Security and Trust*. IEEE, Paris, France, 137–144. https://doi.org/10.1109/PST.2012.6297930

[29] Jan Pennekamp, René Glebke, Martin Henze, Tobias Meisen, Christoph Quix, Rihan Hai, Lars Gleim, Philipp Niemietz, Maximilian Rudack, Simon Knape, Alexander Epple, Daniel Trauth, Uwe Vroomen, Thomas Bergs, Christian Brecher, Andreas Bührig-Polaczek, Matthias Jarke, and Klaus Wehrle. 2019. Towards an Infrastructure Enabling the Internet of Production. In *2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS)*. IEEE, Piscataway, NJ, 31–37. https://doi.org/10.1109/ICPHYS.2019.8780276

[30] Pille Pullonen, Raimundas Matulevičius, and Dan Bogdanov. 2017. PE-BPMN: Privacy-Enhanced Business Process Model and Notation. In *Business Process Management (Lecture Notes in Computer Science)*, Josep Carmona, Gregor Engels, and Akhil Kumar (Eds.). Springer International Publishing, Cham, 40–56. https://doi.org/10.1007/978-3-319-65000-5_3

[31] Mateo Ramos Merino, Luis M. Álvarez-Sabucedo, Juan M. Santos-Gago, and Francisco de Arriba-Pérez. 2019. A Pattern Based Method for Simplifying a BPMN Process Model. *Applied Sciences* 9, 11 (Jan. 2019), 2322. https://doi.org/10.3390/app9112322

[32] Philipp D. Rohde, Enrique Iglesias, and Maria-Esther Vidal. 2023. SHACL-ACL: Access Control with SHACL. In *The Semantic Web: ESWC 2023 Satellite Events (Lecture Notes in Computer Science)*. Springer Nature Switzerland, Cham, 22–26. https://doi.org/10.1007/978-3-031-43458-7_4

[33] Mattia Salnitri, Fabiano Dalpiaz, and Paolo Giorgini. 2017. Designing Secure Business Processes with SecBPMN. *Software & Systems Modeling* 16, 3 (July 2017), 737–757. https://doi.org/10.1007/s10270-015-0499-4

[34] Robert H. Schmitt, Matthias Bodenbenner, Tobias Hamann, Mark P. Sanders, Mario Moser, and Anas Abdelrazeq. 2024. Leveraging measurement data quality by adoption of the FAIR guiding principles. *tm - Technisches Messen* (2024). https://doi.org/doi:10.1515/teme-2024-0040

[35] Sergey Smirnov. 2011. *Business Process Model Abstraction.* Ph.D. Dissertation. Universität Potsdam. https://publishup.uni-potsdam.de/frontdoor/index/index/docId/5807

[36] Sergey Smirnov, Hajo A. Reijers, and Mathias Weske. 2011. A Semantic Approach for Business Process Model Abstraction. In *Advanced Information Systems Engineering (Lecture Notes in Computer Science)*, Haralambos Mouratidis and Colette Rolland (Eds.). Springer, Berlin, Heidelberg, 497–511. https://doi.org/10.1007/978-3-642-21640-4_37

[37] Lianshan Sun, Jaehong Park, Dang Nguyen, and Ravi Sandhu. 2016. A Provenance-Aware Access Control Framework with Typed Provenance. *IEEE Transactions on Dependable and Secure Computing* 13, 4 (July 2016), 411–423. https://doi.org/10.1109/TDSC.2015.2410793

[38] Christina Tsagkani and Aphrodite Tsalgatidou. 2022. Process Model Abstraction for Rapid Comprehension of Complex Business Processes. *Information Systems* 103 (Jan. 2022), 101818. https://doi.org/10.1016/j.is.2021.101818

[39] Wil M. P. Van Der Aalst. 2011. Intra- and Inter-Organizational Process Mining: Discovering Processes within and between Organizations. In *The Practice of Enterprise Modeling*, Wil Van Der Aalst, John Mylopoulos, Michael Rosemann, Michael J. Shaw, Clemens Szyperski, Paul Johannesson, John Krogstie, and Andreas L. Opdahl (Eds.). Vol. 92. Springer Berlin Heidelberg, Berlin, Heidelberg, 1–11. https://doi.org/10.1007/978-3-642-24849-8_1

[40] Joerg Walden, Angelika Steinbrecher, and Maroye Marinkovic. 2021. Digital Product Passports as Enabler of the Circular Economy. *Chemie Ingenieur Technik* 93, 11 (2021), 1717–1727. https://doi.org/10.1002/cite.202100121

[41] Sascha Welten, Marius de Arruda Botelho Herr, Lars Hempel, David Hieber, Peter Placzek, Michael Graf, Sven Weber, Laurenz Neumann, Maximilian Jugl, Liam Tirpitz, Karl Kindermann, Sandra Geisler, Luiz Olavo Bonino da Silva Santos, Stefan Decker, Nico Pfeifer, Oliver Kohlbacher, and Toralf Kirsten. 2024. A Study on Interoperability between Two Personal Health Train Infrastructures in Leukodystrophy Data Analysis. *Scientific Data* 11, 1 (June 2024), 663. https://doi.org/10.1038/s41597-024-03450-6

[42] Mark D. Wilkinson, Michel Dumontier, I. Jsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E. Bourne, et al. 2016. The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data* 3 (2016), 160018. https://doi.org/10.1038/sdata.2016.18

[43] Hideya Yoshiuchi, Ikumi Inoue, and Hiroki Miyamoto. 2021. Cross-Organizational Data Sharing Technology for Data Management Platform in the Manufacturing Industry. In *Proceedings of the 2021 6th International Conference on Cloud Computing and Internet of Things (CCIOT '21)*. Association for Computing Machinery, New York, NY, USA, 29–35. https://doi.org/10.1145/3493287.3493292