Graph-Based Managing and Mining of Processes and Data in the Domain of Intellectual Property

Gerd Hübscher[a,d], Verena Geist[b], Dagmar Auer[c,*], Andreas Ekelhart[e], Rudolf Mayer[e], Stefan Nadschläger[c], Josef Küng[c]

[a] *Hübscher & Partner Patentanwälte GmbH, Spittelwiese 4, 4020 Linz, Austria*
[b] *Software Competence Center Hagenberg GmbH, Softwarepark 21, 4232 Hagenberg, Austria*
[c] *Johannes Kepler University Linz, Altenbergerstr. 69, 4040 Linz, Austria*
[d] *polymind GmbH, Salierigasse 1/5, 1180 Vienna, Austria*
[e] *SBA Research gGmbH, Floragasse 7, 1040 Wien, Austria*

Abstract

Digitalization of knowledge work in communication-intensive domains such as intellectual property protection poses great challenges but also opportunities to improve today's working environments. The legal domain is strongly characterized by knowledge work, whereby, despite a common legal framework, creativity of individual experts is decisive. This knowledge-intensive work deals with a great amount of data objects, not only as a working basis, but also as a result. While experts heavily follow individual working styles, they still rely on a vast amount of administrative tasks, which are carried out by the supporting staff. These tasks are expected to be performed regularly, reliably and without errors, despite necessary adjustments to the current case and the changing legal framework. Today, knowledge work and administrative tasks are typically supported by different tools that are hardly integrated. Therefore, the tracing of continuous work processes based on exchanged data objects is a great challenge. This traceability is crucial, not only for legal security reasons, but also to enable mining and learning of applicable knowledge about processes. In this paper, we propose a bottom-up approach, which applies a continuously evolving graph of integrated data objects and tasks to model and store static and dynamic aspects of administrative as well as knowledge work, and test the approach in a real-world setting in the domain of intellectual property. We further present initial results of a novel dependency-based mining approach to learn data-dependent task sequences in the graph-based model and discuss several methods for enabling privacy-preserving sharing and mining.

Keywords: Graph-structured Data, Business Process Management, Knowledge Work, Process Mining, Graph Mining, Privacy
2020 MSC: 68T05, 68T30, 68U35

*Corresponding author
Email address: dagmar.auer@jku.at (Dagmar Auer)
URL: www.huebscher.at, www.polymind.gmbh (Gerd Hübscher), www.scch.at (Verena Geist), www.jku.at (Dagmar Auer), www.sba-research.org (Andreas Ekelhart), www.sba-research.org (Rudolf Mayer), www.jku.at (Stefan Nadschläger), www.jku.at (Josef Küng)

Preprint of doi:10.1016/j.is.2021.101844, Journal of Information Systems

1. Introduction

Knowledge-intensive application domains such as the juridical domain are gaining more and more interest in approaches for adequate work support. Well-structured administrative work, which is often about extracting data from documents or transferring data to different data pools and people, can be supported by traditional business process management (BPM) systems [1, 2]. These systems typically rely on well-defined, stable processes with a high number of repetitions and often follow a control-flow oriented approach. In contrast, knowledge work [3, 4, 5, 6] is more about managing and applying knowledge to creatively build new knowledge, often in a highly dynamic environment, where legal constraints and compliance rules are frequently changing. In an organizational setting, however, different kinds of work must go hand in hand. Only a flexible, adaptable and at the same time very well coordinated and comprehensible interaction makes an organization successful.

From a knowledge perspective, observable communication data is mostly unstructured and requires a-priori knowledge to extract semantic concepts. The lack of a model that allows to define and handle *mental models*[1], which typically evolve during daily work, hinders the development of a consistent process of data transformation tasks of individual users. This might be one reason that business process modeling, as well as individual and organizational learning processes, have not yet been successfully applied to data-driven, process-oriented knowledge work.

A further complicating factor is that the externalization of tacit knowledge is a lengthy process, which requires the repetitive articulation (converting tacit knowledge into explicit knowledge) and internalization (using that explicit knowledge to extend one's own tacit knowledge base) [7]. We have observed that externalization of tacit knowledge regardless of a specific use case is particularly difficult for knowledge workers and requires means to systematically capture work artifacts and facilitate externalization of derivable knowledge by way of abstraction.

From a process perspective, the need for a *high degree of adaptability* for non-routine, problem-solving tasks does not fit with traditional activity-centric BPM systems, which mainly adopt a top-down approach for predefined administrative processes and particularly lack the integration of data [8].

In an organizational setting, not only highly dynamic knowledge work, but also well-structured standardized processes within the same context need to be supported, e.g., legal constraints or compliance rules that clearly define procedures. Furthermore, traceability of actions and decisions based on the collected information is increasingly becoming a mandatory requirement. The obligation to provide information under the General Data Protection Regulation and support for providing evidence e.g., in the course of reinstatement cases or unplanned takeover of tasks due to sick colleagues, are everyday challenges in today's organizations.

In this article, we propose a highly adaptable, integrated model, which considers data and tasks equivalently and supports their interaction. Data are described on different levels of granularity, ranging from a complex patent portfolio, a case file, an e-mail,

[1] Despite of different definitions of the term mental model, we consider a mental model as an abstract representation of a certain thing or set of things, such as people, objects, places or actions that can be organized in hierarchies.

a natural person, a deadline, an e-mail address, to a single string. These entities are organized in a directed graph, with edges indicating containment or association. To allow for a high level of adaptability and continuous evolvement of the overall model, we consider a data-centric, multi-layer model. Based on a stable meta model, concepts for types and instances can be adapted or added at runtime. Duplicated data objects should be avoided to provide consistent, high-quality data, which provide the basis for executing and tracing data-driven processes. The execution of a task is determined by data and produces data [9, 10, 6]. These (intermediate) results are therefore the driving force of the business processes and form sequences of alternating tasks and data artifacts that evolve in the graph when knowledge is created.

Process instances in general, and specifically in the legal domain, carry information about potentially confidential cases, as well as information about individuals partaking in those processes. Protecting confidential information about cases and clients is a fundamental requirement of law firms and prescribed by law. Therefore, in case of non-compliance, law firms may face high fines and further negative impact due to reputation damage. Thus, utilizing personal and confidential data poses a challenge to collaborative process graph sharing and mining. We discuss potential methods for enabling privacy-preserving sharing and mining, and describe already implemented approaches.

Our contributions to solve the identified problems are (i) a highly flexible and adaptable three-layer graph-based model with a stable meta model which integrates data and task management conceptually. The high level of adaptablility is achieved by a data-centric domain model and an instance model which allows to support highly creative, flexible knowledge work and guided administrative work, including detailed tracing of task and data flows, (ii) a real-world demonstration of our model in the domain of intellectual property, (iii) a discussion on methods to preserve privacy and confidentiality in our model, and (iv) initial results from applying a mining approach in order to learn from task sequences carried out in the integrated data and process graph and to assist users by suggesting next process steps.

In Section 2, we outline the research methodology and provide our research questions. The following Section 3 gives an overview of related work in the area of knowledge work and flexible support for business processes. Our proposed model, the modeling approach and the prototype implementation are presented in Section 4. In Section 5, we discuss methods to preserve privacy and confidentiality in our graph-based model for sharing and mining scenarios. In Section 6, we introduce a data dependency-based mining approach that learns from recorded task sequences in the proposed model and is able to recommend a selection of probable next process steps to the user. Finally, in Section 7, we summarize and discuss the main results. The paper concludes with a summary and an outlook on future work in Section 8.

2. Research Methodology

We follow a *design science research (DSR) methodology* [11]. Vom Brocke *et al.* [12] describe DSR as "... a problem-solving paradigm that seeks to enhance human knowledge via the creation of innovative artifacts."

The following Fig. 1 provides an overview of the applied DSR framework, which is based on [11, 12]. Starting with the definition of needs based on a real-world problem in a defined environment, artifacts and knowledge are designed, developed and evaluated in

an iterative research process. Evaluation results are used in further iterations to improve the design and implementation. Furthermore, the achieved results are applied in the appropriate real-world environment, and design knowledge (i.e., "... knowledge of how things can and should be constructed or arranged (i.e., designed)" [12]) is added to the knowledge base.

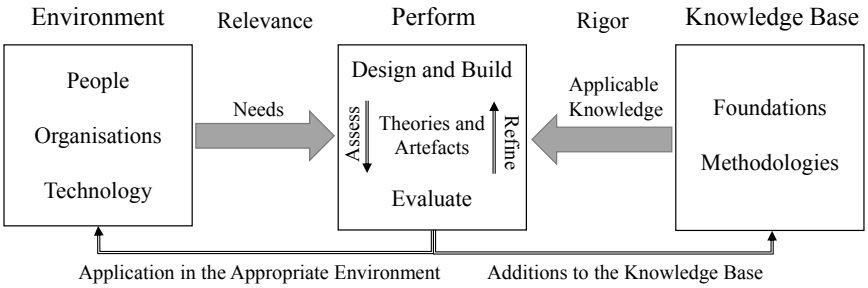


Figure 1: Applied DSR framework (cp. [11, 12])

The research results presented in this paper were achieved in a series of joint research projects with partners from business and research, starting in 2017. Within each of these projects, we followed a design science research methodology. The initial business partner was a patent law firm of one of the team members, who is also a computer science researcher and developer. After the initial project, the polymind GmbH was founded to ease product development and to enable and promote practical application of the promising basic concept in the real world. However, the starting point of our research in the domain of patent and trademark prosecution remained the main environment throughout the projects. In the following, we give a summarized overview of the research efforts relevant for the results we present in this paper.

The environment of our research in the domain of patent and trademark prosecution is characterized by a highly standardized, legally prescribed process on the one hand, and individual knowledge work, for example, when translating new technical knowledge into legally binding language on the other. Furthermore, this work requires intensive communication, within an organization as well as with clients and external partners. Two organizational roles are the most important in this environment – office administration and patent attorney. Work of people in these roles is strongly interconnected, which is not sufficiently supported by the current IT systems, because they lack integration of processes and data. Today, a range of rather isolated tools is used, i.e., a traditional customer relationship management system, a document management system, an email system and not to forget about folders and documents in the file systems of the local computers as well as the server. Traceability of the overall process and data is not only important with respect to the General Data Protection Regulation, but also concerns the all due care requirement with respect to the re-establishment of rights. Furthermore, the opportunity for individual and organizational learning from previous work requires adequate support. Not all of these needs were considered within the first project, but the relevant environment was continuously extended to iteratively develop new ideas and results on top of evaluated previous ones. In particular, the lacking integration of processes

and data in traditional activity-centric BPM systems is regarded as an essential point, preventing their successful use to support knowledge-intensive processes [8]. Therefore, we consider a data-driven approach promising, which equally considers data and tasks. Furthermore, to support highly creative, individual knowledge-intensive work on the one hand, and well-structured, predefined, highly repetitive work on the other, there is a strong need for a highly flexible, adaptable model. Continuous evolution of the model should allow to overcome the distinction between design time modeling and runtime execution of processes. To formalize and encode domain-specific, explicit knowledge, some kind of shared knowledge base is needed, which defines the task concepts including relevant data. To increase this knowledge base, also implicit knowledge of the users needs to be included, e.g., by learning from the way people perform their daily work. As often sensitive personal data is involved in patent and trademark prosecution, privacy aspects must be considered as well.

Based on the selected needs the core research is performed, considering the following research questions:

RQ1: How can a highly accurate knowledge representation be obtained to support well-structured, predefined data-driven processes as well as highly adaptable, individual ones?

RQ2: How and to what extent can sensitive information inside the graph-based model be protected for data sharing and process mining?

RQ3: How and to what extent can statistical, process and graph mining methods support the discovery of data-dependent sequences in the given context and facilitate predictive analytics, such as suggesting the best possible next tasks to a user?

Each of these research questions was studied in an iterative design, build and evaluation process based on the needs and relying on knowledge from disciplines such as knowledge work, BPM, knowledge-intensive business processes, information management, knowledge management, process mining, privacy and graph data modeling. Some specific parts of the initial knowledge base, especially the foundations, are discussed in the state of the art in Section 3.

For the evaluation phase we used different methodologies, depending on the research question and the focus of the particular iteration. In the initial iterations, we focused on analytical proofs of concept, further involving external experts into feedback circles and discussions, followed by a proof by construction to show that the concept can be implemented appropriately. The prototype was further used to conduct a case study on real-world data, which were imported from the management systems in place. The participants of the case study performed certain predefined tasks after some introduction to the prototype, but also played around with it individually. Besides some observations and studying the produced results, also interviews with the people participating in the case study were performed to get feedback on the results but also to generate new ideas for the following iterations. We tested the case study with one law firm and then performed the case study with five law firms in comparable environments. To study more technical aspects also different metrics were selected, defined and evaluated. Since real-world data can sometimes lead to privacy problems and some ground truth was needed to reduce the data volume and to deal with long-running processes, we also used test scenarios with

generated test data (according to real-world scenarios) for evaluation, especially in the context of research question RQ3.

To sum up, throughout these research projects several concepts, models and prototypes were built and evaluated. The case study not only convinced the participating offices, but also triggered new application possibilities which were further refined and developed at polymind GmbH, especially in areas such as learning management, and research and development support. The results, i.e., the concepts, models and artifacts, added to the knowledge base are described in detail in Section 4 to Section 6, followed by the most important findings from answering the research questions in Section 7 (i.e., RQ1 in Section 7.1, RQ2 in Section 7.2 and RQ3 in Section 7.3).

3. Related Work

The related work sketched in this section is concerned with the overall topic of this work, i.e., integrated data and process model. Further literature on aspects, such as privacy or mining, are included in the proper sections.

Business process management (BPM) [1, 2] solutions are already state-of-the-art for well-structured, predefined standard tasks and processes with a high number of repetitions, like typical *administrative tasks*. The underlying models are usually activity-centric and control flow-oriented. Predefined process models with traditional BPM approaches restrict flexibility but provide good guidance for the users, clearly indicating how things should be done. These models are not intended to be constantly adapted by users. Further, business process modeling languages such as BPMN[2] do not focus on data. This aspect is rather regarded as one of their weakest points [13, 14]. BPMN, for example, provides so-called data objects to document data usage. These data objects are unstructured and have no execution semantics. Thus, neither the required level of integration of data and tasks [15], nor an adequate representation of complex data objects can be achieved. With the increasing importance of *knowledge work*, which is characterized by its adaptable and creative nature [3], no longer only well-structured processes with highly repetitive tasks must be considered, but highly flexible and easily adaptable emergent, collaborative ones [6, 16]. The characteristics of such *knowledge-intensive business processes* in detail differ due to the heterogeneous application domains of knowledge work - from highly-creative, non-repeatable, completely unpredictable work to areas with constraints and rules (e.g., legal frameworks or compliance rules) but still a high level of individual, knowledge-relying work. Di Ciccio *et al.* [8] discuss a comprehensive set of characteristics of knowledge-intensive processes, additionally considering goal-oriented and event-driven.

Traditional *activity-centric BPM* approaches were extended to allow for more flexibility at runtime, e.g., via process configuration, variants, or ad-hoc tasks. Still the frame of their stable build-time models is restrictive and data is hardly considered [17, 18].

Thus, *data-driven models* started to get more attention during the last 15 years, especially in research. The key driver for process execution is no longer a predefined control flow, but the availability of data. Case management [19, 9, 20, 17], for example,, allows to define quite flexible data-driven models, but only coarse-grained data are considered.

[2]Business Process Model and Notation, www.omg.org/spec/BPMN

Object-aware processes [21, 22] better support the data aspect, but are not designed for adaptability and dynamic model evolution at runtime.

Other approaches such as *rule-based* or *constraint-based declarative models* [23, 17] offer a higher level of flexibility at design-time, but many of them generate control flow-oriented process models, e.g., SDeclare [24] which rarely take the data perspective into account.

None of these approaches sufficiently integrate the data and task view. Especially, fine-grained, emergent data, information and knowledge are not within the focus.

Previous works towards a more flexible business process technology integrate processes with data and user interaction modeling [15, 25]. They base on a methodology to model interactive software systems with form-based, submit/ response-style interfaces [26]. This basis provides a clear semantics of dialogues, constituted by application programs and bridging process states, as typed, bipartite state machines called *formcharts* [27].

In [15], BPMN is extended with submit/response-style user interaction modeling to mitigate communication problems between business analysts and software engineers, having different views on a system. Thereby, a two-staged interaction schema is enforced, which consists of providing a page (= information) to a user and processing submitted data. In [25], formcharts are further extended to the needs of business process specification, i.e., to support users/roles, the worklist paradigm, and parallelism. The resulting typed *workflow charts* are proposed as a new formalism for modeling and automating business processes [28]. They also build a solid foundation for the design of an integrated business process platform in [29].

An experience-proven methodology for modeling interaction with web information systems as task-oriented systems is presented in [30].

There are many similarities but also key differences between the presented previous works and the current model proposed in this work: (i) all approaches offer a rigorous way to specify information systems. Draheim and Weber consider certain kinds of submit/response-style systems [27, 26], whereas Schewe and Thalheim address a broader field of web information systems [30]. Both methodologies aim to design human/computer interaction in workflow-intense systems. However, our focus is on knowledge presentation and communication of human workers in dynamic, knowledge-intense environments.

(ii) Data models play a central role in all works. The work in [30] is characterized by a tight coupling with databases. Similar to our work, a layered data model, basically divided into an *information* model (of aggregated, processed data) and an unchangeable *data*/message model (of observable data) is used in [27], following the basic understanding of the two lowest levels of the DIKW pyramid [31].

(iii) All approaches make use of directed graph-based models. Formcharts, for example, exhibit a bipartite structure in state transition diagrams; similarly we use a bipartite model for storing data dependencies in a graph database. Unlike the other approaches, we do not link data (and information) objects solely in a static sense but also through experience and use (cp. definition of *knowledge* in [31]).

(iv) Data and information are typed in all works, which allows for constraint specification. However, we cannot use a strict schema because we want the model to evolve dynamically at runtime (driven by the user). For example, in [27] and all formchart-based approaches, a user action is a method call with clearly defined input parameters (which

represent input capabilities of a form), whereas in the TE𝔸M model, a user action is a domain-specific task, which follows a flexible type to reference and create data objects and allows knowledge workers to extend or build new types during their work.

(v) Finally, all approaches can be used as conceptual system modeling languages (e.g., for redocumentation purposes) but also as executable specification languages (towards business process automation). A major difference to our work is that the proposed approaches in [15, 25, 28, 30] are intended to model and automate a-priori known, structured processes (e.g., using BPMN [13, 29]), whereas due to the creative nature of knowledge work little to nothing needs to be predefined in the TE𝔸M model.

Earlier works in the area of *semantic BPM* [32, 33] focus on combining Semantic web services with BPM technology to support agile process implementation, but with a rather technical perspective on machine-accessible semantics, not sufficiently respecting dynamic processes with different participants. To gradually develop fine-grained types and instances from different data sources (structured or semi-structured data as well as free text), knowledge representation such as *knowledge graphs* [34, 35] are promising. Even though the *Resource Description Framework (RDF)* [36] is well-established, the absence of (i) attributes on vertices and edges and (ii) unique identities for relationships hinders an application for our problem case.

4. The TE𝔸M Model

The overall vision for the TE𝔸M model (TE𝔸M – in**TE**grated d**A**ta and t**A**sk **M**ultidimensional graph) is to support people to effectively and efficiently work on their predefined administrative tasks as well as on flexible knowledge-intensive ones. Thus, we propose a highly flexible model of integrated data and tasks, which allows for seamlessly maturing of the model also at runtime. With regard to process-awareness, we follow a data-driven approach, i.e., tasks can be activated it the needed data is available. While performing work, task-related as well as the data-related information is continuously tracked, which allows for fine-grained tracing of processes and data flows as well as the evolution of data.

The required adaptability and flexibility of the overall model is achieved via a three-layer architecture.

4.1. Three Layers Architecture

The architecture of the TE𝔸M model consists of the three layers, i.e., the *meta model*, the *domain model* and the *instance model*. Each of these layers supports the dimensions *data* and *task* as well as their integration. Each of these models is described in a graph model. Fig. 2 provides an overview of these models and their dependencies.

The *meta model* defines the stable foundation of the TE𝔸M model - the core characteristics of both, the instances (i.e., data object, data object relation, task, and task relation) and their classifying types (i.e., data object type, data object type relation, task type, and task type relation). Data object types can be related to each other via data object type relations, which allows to build fine-grained data views in the domain model. This pattern also holds for the instances. As we follow a data-driven approach to describe the behavior of the system, the interface of each task type (and thus task) is defined by task type relations (task relations) to or from the corresponding data object

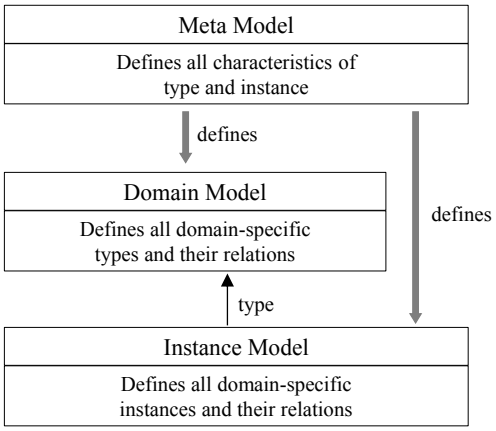


Figure 2: Overview of the three layers of the TEAM system.

types (data objects). Further, data object type relations (data object relations) can be a result, an output, of a task type, but it must not be an input. Such fundamental model constraints are also defined in the meta model.

The *domain model* holds all domain-specific types. They do not correspond to types in programming languages, but are mental models [37] used for classifying instances. These types can be based on an established ontology of the legal domain and can be continuously refined at runtime.

The *instance model* describes all instances, which are classified by a domain-specific type (see arrow with label *type* in Fig. 2).

Seamless knowledge and process maturing can be achieved by starting with only a small set of basic types and instances, and adapting and extending the domain model and instance model whenever needed. Therefore, we do not rely on rather stable data types, but on a flexible approach using types for classification.

We use the following real-world scenario to explain the details of the TEAM model within some practical context.

4.2. Real-world Scenario

The following scenario shows a small process in a patent office including administrative and knowledge-intensive work.

Starting point is the receipt of an e-mail from a client with two distinct concerns: (1) It contains the notification that the address of one patent proprietor has changed. (2) The client requests whether, in view of the first office action received, it is appropriate to pursue a certain patent application. The reply is urgently expected because the time limit for filing observations in reply to the office action expires in a few weeks. This e-mail initiates two actions: (1) As it cannot be assumed that the e-mail is already an order to enter the address change in the relevant official registers, certain facts need to be clarified: property rights where the indicated patent proprietor is the owner, costs of the changes and finally a professional assessment whether such a change should be indicated before the relevant offices at all. Furthermore, this change potentially not only

affects a single case but several ones. (2) Requires to study the related case file as well as the office action, to procure the prior art cited therein, to exam the citation, and to elaborate one or more means of defense. Experience and knowledge of the practice of the patent office are needed for this. Furthermore, it is questionable whether the stated urgency applies at all, whether extensions of time limits are possible or whether alternative prosecution routes are available. Finally, an evaluation of the existing possibilities and a concrete recommendation for action are required. Since all actions can have serious legal consequences, in particular tracing of actions and decisions based on the collected information is needed. Throughout the discussion of the TE𝔸M model in this section we will focus on different aspects of this scenario.

In the following, we will discuss the core components of the data and the task views in detail and then head towards their integration. We use Latin letters for instance and Greek ones for type abbreviations.

4.3. Data View

In practice much information and knowledge is stored in unstructured or semi-structured documents, thus, we strive to extract a fine-grained representation of the data and classify it with the according type. To increase data quality and to build consistent process chains, each entity is unique within the system, e.g., if several people have the same first name, it is only created once and then the unique entity is referenced by all concerned persons.

The data view defines a graph of fine-grained concepts for classifying data entities (*data object type,* δ) and their relations (*data object type relation,* ρ), the type model of the data view. The instance model holds the fine-grained data instances and coarse-grained documents (*data object,* d), as well as the relations (*data object relation,* r) between them.

Definition 1. – *A* ***data object (d)*** *represents the mental model for an instance, e.g., the natural person* `John Doe`*. Each data object is unique within the TE𝔸M model. Two different kinds of data objects are distinguished -* ***observable*** *and* ***non-observable*** *data objects. An observable data object contains an observable value, while a non-observable data object is the root for the subgraph describing the overall data object.*

Example. The natural person `John Doe` is a non-observable data object as it represents the mental model without providing an immediately observable representation. In contrast, the data object `John` of type `String` is an observable data object as it holds the string value `John`.

Definition 2. – *A* ***data object type*** *(*δ*) describes a concept that classifies a set of data objects, e.g., natural person, address.*

Data objects and data object types are further described by attributes. Data object type attributes define characteristics such as the name of the type or rules to define constraints shared by all data objects classified by this type. Data object attributes differ as they deal with runtime aspects such as status history or the value of observable data objects, i.e., the transactional data. Partonomical relationships are not described by attributes, but by relations. Therefore, entropy decreases along the direction of these relations.

Definition 3. – *A **data object relation (r)** can be either a partonomical (with the kinds **has** and **hasValue**) or an associative (with kind **role**) relation (thus directed) between two data objects.*

Non-observable data objects are always connected to their containing, superordinate data objects via data object relations of kind *has*, *observable ones* need the kind *hasValue*. A data object relation of kind *role* links a data objects to its role. The direction of the relation is from role to object, i.e., the semantics of *isRoleOf*. The corresponding concept on the type level is the data object type relation. The partonomical relations of kind *has* and *hasValue*, the associative of kind *role* as well as the generalization via the *is relations* are used to build the concept hierarchy.

Definition 4. – *A **data object type relation** (ρ) is a bipartite, directed type-level relation, which specifies its kind (taxonomical, partonomical or associative) and its source and target data object type.*

Data object relations are classified by data object type relations, thus each data object relation holds a reference to its classifying data object type relation in its attribute `type`.

In the following we demonstrate the defined concepts with a real-world example. An *observable data object* $d_i^\star$ has exactly one unique value that can be observed e.g., in messages. Its value can be of any primitive data type such as String or Integer, but also represent a whole document. For example, the value of $\mathrm{d}_1^\star$ is `"John"` in Fig. 3. $\mathrm{d}_1^\star$ is an observable data object with the data type `String`. *Non-observable data objects* d_i° have no value. They stand for abstract mental models which are represented (encoded) by related observable but also non-observable data objects (e.g., d_4° in Fig. 3). Both, observable $\mathrm{d}_i^\star$ and non-observable d_i° data objects are discrete, unorganized and have no specific meaning. The two sets of data objects are defined in equation (1).

$$D^\star = \{\mathrm{d}_1^\star, \ldots, \mathrm{d}_n^\star\} \quad D^\circ = \{\mathrm{d}_1^\circ, \ldots, \mathrm{d}_n^\circ\} \quad D = D^\star \cup D^\circ \tag{1}$$

Meaning is added by the relations. We can partition the edges R into two sets $R^\star$ and R° (cf. equation (2)).

$$R^\star = \{\mathrm{r}_1^\star, \ldots, \mathrm{r}_n^\star\} \quad R^\circ = \{\mathrm{r}_1^\circ, \ldots, \mathrm{r}_n^\circ\} \quad R = R^\star \cup R^\circ \tag{2}$$

$R^\star$ connects observable data objects $D^\star$ and non-observable ones D°, while R° only connects non-observable ones D°. These disjoint independent sets contain the vertices D (i.e., data objects) of a directed graph $G_D = (D, R)$, interconnected by the edges R (i.e., data object relations).

The graph G_D in Fig. 3 describes the non-observable data object d_{10}° (email from `John Doe` with all its details). Starting with the observable data objects $D^\star$, a bottom-up approach is used to build enriched data objects along a path of aggregating *hasValue* and *has* relations as well as the associating *role* relations (e.g., r_3°). Thus, each element is finally assigned to an observable data object ($\mathrm{d}_1^\star$ – $\mathrm{d}_4^\star$), which relates to one or more non-observable data objects (d_1° – d_{10}°). The values of the observable data objects are also provided in Fig. 3.

To support the continuous further development of the data instance model represented by data objects and data object relations all of them have instance attributes such as

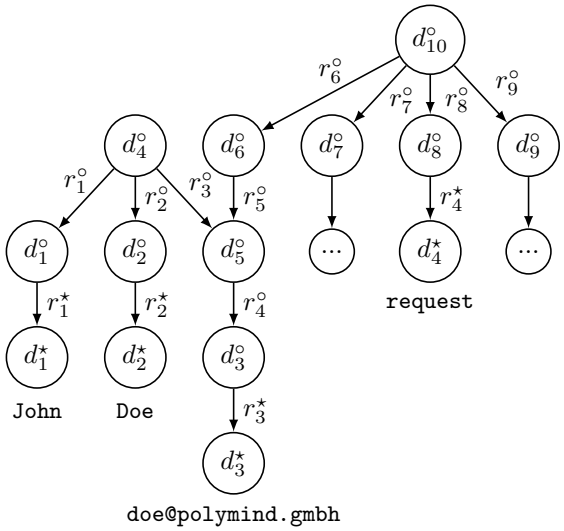


Figure 3: $G_D = (D \cup R)$ with observable $d^\star_i$ and non-observable d°_i data objects.

valid and *invalid*, indicating the current status of the instance as well as the timestamp, when the status has been set to a certain value. Initially data objects and data object relations are *valid*, but they can be *invalidated* and *validated* whenever necessary.

Within the TEAM model, we have two connected semantic subgraphs, the instance and the type graph, integrated via type relations. To maintain an overview, Fig. 3 only shows the data instance graph.

4.4. Task View

Besides the data objects and their relations, the tasks and the relations between data objects or data object relations and tasks build the overall integrated, dynamic model.

Definition 5. – *A **task (t)** represents an instance of an atomic activity of information processing classified by some task type.*

Definition 6. – *A **task type (ξ)** describes a concept that classifies a set of tasks, e.g., extract address information from letter.*

Similar to the data perspective, all entities of the task perspective have attributes at type and instance levels. To specify the data interfaces of tasks (i.e., the data consumed and produced), task type relations are used to define the incoming and outgoing data object types for a task type.

Definition 7. – *A **task type relation (υ)** is a directed type-level relation between a source and a target type with a specific kind [k] (e.g., specifying data dependency, validity of data objects or relations, user action on the task, etc.). Source and target types are mutually a data object type or data object type relation and a task type. There is only one special case, a data object type relation can only be the target of a task type relation and never the source.*

Table 1: Details for the data objects d in Fig. 3.

id	data object type	value
$d_1^\star$	`String`	"John"
$d_2^\star$	`String`	"Doe"
$d_3^\star$	`String`	"doe@polymind.gmbh"
$d_4^\star$	`String`	"request"
d_1°	`FirstName`	-
d_2°	`LastName`	-
d_3°	`EMailAddress`	-
d_4°	`NaturalPerson`	-
d_5°	`EMailContact`	-
d_6°	`Sender`	-
d_7°	`Receiver`	-
d_8°	`Subject`	-
d_9°	`DateReceived`	-
d_{10}°	`EMail`	-

Definition 8. – *A **task relation (y)** is a directed relation between a source and a target with a specific kind. Either the source is a data object and the target a task or the source is a task and the target a data object or data object relation. A task relation is the counterpart of a task type relation on the instance-level and has a task type relation assigned.*

As described in more detail by means of the example in Fig. 4, within the context of a task, task relations describe the usage and creation of data objects as well as data object relations. Task relations thus associate tasks with the information used or generated in them. In this way, task relations also represent the transformation of information that takes place in a task, which is mapped by incoming and outgoing data objects and data object relations. Each task relation represents a traceable action (e.g., with the kind *creates*, *validates* or *invalidates*), which is abstracted on the type level and, thus, becomes available as generally available knowledge about the relevant task type.

Like the data view also this dynamic, process-oriented part of the model is steadily maturing during use.

Equation (3) provides the sets for the task perspective on instance level and for the overall instance graph G_I – data and task perspectives.

$$T = \{\mathrm{t}_1, \ldots, \mathrm{t}_n\} \quad Y = \{\mathrm{y}_1^k, \ldots, \mathrm{y}_n^k\} \quad G_I = (T \cup D \cup Y \cup R) \tag{3}$$

The data and the task views on instance level are integrated via task relations of different kinds k, which can be split into three distinct groups: (1) input and output data objects (with output also data object relations) related to tasks, (2) user assignment to tasks, and (3) status transitions of tasks. These relations need to be defined on the type level in advance.

Fig. 4 provides a small part of the integrated data objects and tasks of the scenario (cf. Section 4.2). For better understanding, we focus exclusively on the instance model.

Even though the type level is not shown, each element of the instance model is associated with its corresponding type.

The scenario describes the user d°_{u1} who reads an email, realizes that the email is relevant for a case d°_{11} and thus picks the case and attaches the email to it. Therefore, a couple of relations are created to document this process. User d°_{u1} first *instantiates* (i) the task t_1 to handle the email d°_{10}. Thus, the corresponding task relation y^i_1 from the user to the task is created. Furthermore, the email d°_{10} is linked to the task as an incoming data object via the task relation y^r_4. Then user d°_{u1} *allocates* (a) the task and the task relation y^a_2 is added. Finally, user d°_{u1} *processes* (p) the task t_1 and another task relation (i.e., y^p_3) is added to the graph. To link the incoming email d°_{10} to the relevant case file d°_{11}, the case file needs to be added as an incoming data object to the task via a task relation (i.e., y^r_5). Then the user links the email to the case file by creating a data object relation from the case file to the email (i.e., r°_{11}). To document that this relation has been created in the context of task t_1, the task relation y^c_6 is added. Table 2 provides an overview of the types of all instances involved in Fig. 4.

The latter part of the scenario also shows the key concept for supporting *traceability*, i.e., to be able to explain at any time *why* objects are related to each other. Therefore, edges also have properties, such as timestamps, status information, etc., which can be target of another edge, thereby creating, referencing or invalidating the relation. This requires extending the classical understanding of graphs as nodes connected by edges by also allowing 'edges connected by edges'. Note that we do not use hypergraphs but rather an artificial node of type 'edge', which is simply represented as an edge only. Thereby, the incoming/outgoing edges remain the same and our extended view on graphs can be mapped to RDF (via reification). Thus, equivalence to a classical graph is ensured.

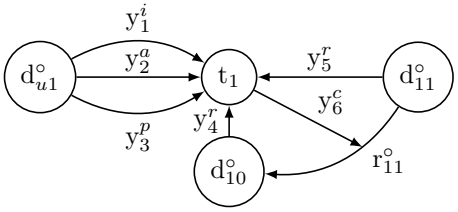


Figure 4: Graph G_I, integrating instances of the data and task perspectives.

Task relations also denote the exchange of data objects between tasks, thereby describing the underlying business processes and make communication between users explicit. Thus, the orientation of the corresponding *task type relations* explicitly documents the process direction.

4.5. Data-dependent Task Sequences

As each data object is unique within the TEAM model, the relations between data objects and tasks result in a continuous, integrated instance graph G_I also describing all business processes implicitly.

To prepare for our approach to mining these highly adaptable data-driven business processes, we present a part of the real-world scenario in more detail (cf. Section 4.2). In Fig. 5 more tasks and users are considered. Different colors are used for data object

Table 2: Details for the instances in Fig. 4.

id	type
t_1	`ProcessEMail`
d°_{u1}	`User`
d°_{10}	`EMail`
d°_{11}	`CaseFile`
r°_{11}	`CaseFileHasEMail` of kind `has`
y^i_1	relation of kind `instantiates`
y^a_2	relation of kind `allocates`
y^p_3	relation of kind `processes`
y^r_4, y^r_5	relation of kind `references`
y^c_6	relation of kind `creates`

relations and task relations to improve readability. The types for all elements involved in the scenario are summarized in Tab. 3. The scenario in Fig. 5 contains the two users d°_{u1} (backoffice) and d°_{u2} (patent attorney) working on three tasks t_1, t_2, and t_3. Within the context of task t_1 user d°_{u1} creates a meeting act d°_1 and a deadline d°_2 which is then connected with the meeting act by r°_1. Users and tasks are connected by task relations of different kinds, e.g., instantiates (y^i_1), allocates (y^a_1), and processes (y^p_1). The results of t_1 are the inputs of t_2. Thus, when user d°_{u2} is informed that there is an assigned deadline d°_2, the user creates a new task t_2 and works on it to produce the resulting meeting notes d°_3 related to the meeting act d°_1 via r°_2. As it is the backoffice's task to prepare the offer d°_4 and send it to the client via e-mail d°_5, d°_{u2} instantiates the corresponding task t_3 referencing the relevant data object d°_3. Then d°_{u1} takes over this task t_3, creates the offer document d°_4 and attaches it to the e-mail d°_5.

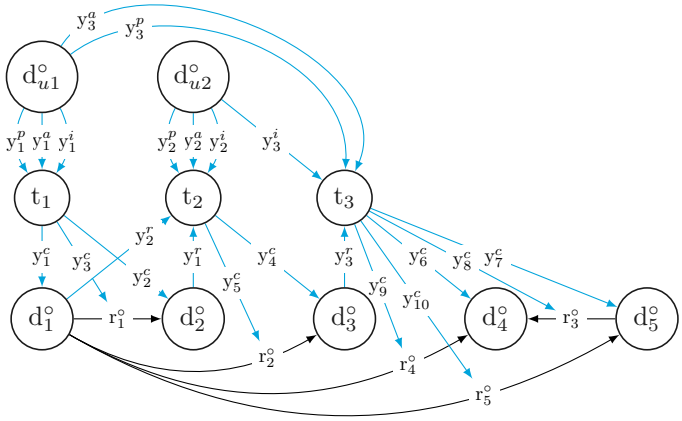


Figure 5: Graph G_I, integrating instances of the data and task perspectives.

Data objects not indicating users are related to tasks either via task relations of kind creates y^c_n or references y^r_n. Task relations linking users to tasks are of kinds such as instantiate y^i_n, allocate y^a_n or process y^p_n.

Table 3: Details for the instances in Fig. 5.

id	type
t_1	`CreateMeetingAct`
t_2	`ConductMeeting`
t_3	`PrepareOffer`
d°_{u1}, d°_{u2}	`User`
d°_1	`MeetingAct`
d°_2	`Deadline`
d°_2	`MeetingNotes`
d°_2	`Offer`
d°_2	`EMail`
r°_n	data object relation of kind `has`
y^i_n	task relation of kind `instantiates`
y^a_n	task relation of kind `allocates`
y^p_n	task relation of kind `processes`
y^r_n	task relation of kind `references`
y^c_n	task relation of kind `creates`

To sum up, all relations between user and task are documented via task relations. Task relations also denote the exchange of data objects between tasks, thereby describing the underlying business processes and making communication between users explicit. Thus, the orientation of the corresponding *task type relations* explicitly documents the process direction.

4.6. Backend and User Interface Prototype

The prototype for the TEAM System has been developed relying on established design patterns in enterprise systems and latest ones in knowledge processing systems (e.g., [38, 39]).

The TEAM System is divided into a server (backend) and a client (frontend). The server is implemented using the Java Spring application framework, especially Spring-Boot[3] and the multi-model database ArangoDB[4] for storing the graph data. We deliberately chose a multi-model database, where an edge has the same base type as a node (i.e., document) and connects elements of the more general type document, in order to allow 'edges connected by edges' – enabling traceability – in a simple and straightforward way. The frontend is developed in Angular 8 [5] using the framework's extensive support for reusable, yet dynamic interface components, which allows to adapt the user interface to the evolving, underlying graph.

The basic architecture of the server is a three-layer architecture: interface, service and data access (see Fig. 6). The communication interface with the client provides functionality via controllers, based on the REST architecture style [40]. The second

[3] `https://spring.io/projects/spring-boot`
[4] `https://www.arangodb.com`
[5] `https://angular.io`

layer deals with business services, while the third layer applies the *data access object* (DAO) design pattern [41] to access data in ArangoDB.

User interaction is task-oriented. Therefore, the work context of the users is their individual tasks with the task contexts, i.e., the connected data objects. Thus, users can focus on what they want to do, and get assistance in identifying data, information and knowledge they need. Users can add new data and relations whenever needed in the context of specific tasks, thus, user interaction builds the evolving graph model following the layered approach.

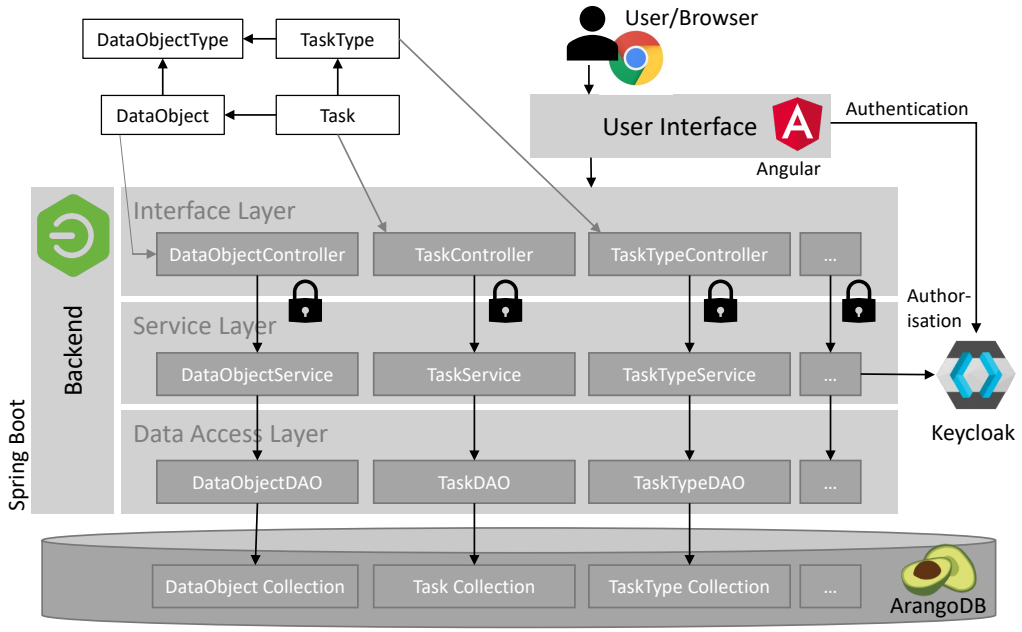


Figure 6: Overview of the system architecture.

5. Privacy

Process instances in general, and specifically in the legal domain, carry information about potentially confidential cases, as well as information about individuals partaking in those processes. Protecting confidential information about cases and clients is a fundamental requirement of law firms, and typically secured by non-disclosure agreements. Therefore, in case of non-compliance, law firms may face high fines and further negative impact due to reputation damage.

Personally identifiable information (PII) are any data that could be used to identify a particular person. Common examples include a full name, a social security number, document numbers (passport, driver's license), e-mail addresses or telephone numbers. The Health Insurance Portability and Accountability Act of 1996 (HIPAA)[6] lists several

[6] https://www.cdc.gov/phlp/publications/topic/hipaa.html

further potential PIIs. Beyond this, datasets in the legal domain may contain other sensitive information not considered PII, such as companies or organizations involved, which needs to be kept confidential for business reasons.

Protecting the privacy of individuals involved in the legal processes is considered a fundamental human right, and thus is included in the legislation of different countries. In the EU, data controllers must design information systems with privacy in mind according to the General Data Protection Regulation (EU GDPR). The knowledge-intensive processes and graph structures introduced in this paper contain detailed information about case activities, involved data, and data subjects (e.g., lawyers, clients, office employees).

There are multiple scenarios that can profit from an in-depth analysis of the data, potentially also in a collaborative setting, and therefore require privacy and confidentiality preserving techniques:

- Analyzing the activity data inside the system could help to optimize processes, but this requires consent from involved individuals. Obtaining this consent can be difficult, considering that process discovery operates in an exploratory fashion without a clear analysis question in mind at that point in time [42].
- In a collaborative setup, law firms could profit by exchanging knowledge models, but they could be reluctant to permit analysis of their process information, fearing that confidential information might get into the hands of potential competitors or other interested parties.
- For testing and development purposes, production data is often beneficial but due to privacy and security reasons its usage is restricted.

5.1. Privacy Protecting Methods

A basic data sensitization approach is the removal of directly identifying attributes, e.g., the ones mentioned above. Some of these attributes might not be removed, but rather replaced with other, random identifiers, in a process commonly referred to as pseudonymization. However, still from such treated data, information can be inferred. For instance, [43] mentions that 87 % of U.S. citizens in 2002 could be re-identified by using attributes zip code, sex and date of birth. These attributes are called quasi-identifiers, as they do not identify by themselves, but may identify in their combination. Re-identification in pseudonymous datasets is often achieved by matching data from the published dataset with other available databases.

Data anonymization approaches address the problem of data protection and privacy / confidentiality aspects, by further sanitizing the data before publishing or processing. Well-known approaches include *k-anonymity* [43] or *differential privacy* [44]. For a detailed overview on privacy-preserving methods, see [45, 46].

k-anonymity aims to ensure that for a given dataset, there are at least k data objects (rows) that are indistinguishable regarding their quasi-identifiers. Differential privacy, on the other hand, is an approach for publicly sharing information about a dataset by describing the patterns of groups within the dataset, while withholding information about individuals in the dataset. It mathematically guarantees that anyone seeing the result of a differentially private analysis will essentially make the same inference about any individual's private information, whether or not that individual's private information is included in the input to the analysis.

k-anonymity and differential privacy are most commonly applied to tabular, relational data. They can thus also be applied to the data that we consider in this paper, if we treat individual data objects as their own relational tables. However, there might be additional inference possible on top of the tabular representation. This is because the interconnections in the graph might provide additional contextual information that was not considered in anonymizing the individual tables.

It has been shown that removing the identity of each node in a social graph before publishing does not always guarantee privacy, as the structure of the graph, combined with prior knowledge of an attacker, could allow the identification of individuals [47]. Motivated by the works [47, 48] the authors in [49] introduced a definition of anonymity in graphs. Feder *et al.* [50] build on previous definitions and propose a formal (k, ℓ)-anonymity algorithm for the graph anonymization problem. A graph is (k, ℓ)-anonymous, if for every node in the graph there exist at least k other nodes that share at least ℓ of its neighbors. They focus on finding the minimum number of edges to be added so that the graph becomes (k, ℓ)-anonymous. Subsequent works on graph anonymization can be found in [49, 51, 52, 53]. Aggarwal et al. [54] examine the problem of node re-identification from anonymized graphs and show that even low levels of anonymization require perturbation levels which are signification enough to result in a massive loss of utility. In the survey [55] one of their conclusions is that state-of-the-art anonymization schemes are vulnerable to several structure-based de-anonymization attacks. Later works also focus on the challenge of heterogeneous graphs, compared to simple networks with only one node and edge type [56, 57].

5.2. Privacy in Process Mining

In the following, we discuss existing anonymization approaches for process mining and possible solutions for the specific data structure and setup we propose in this article.

Until recently, process mining and privacy were considered orthogonal [42], and discovering accurate process models from event logs was the main goal. While the trade-off between privacy and data mining [58] has been illustrated and analyzed before, [59] was one of the first to discuss technical- and organizational privacy challenges for process mining. A requirement for anonymization techniques in process mining is to have an acceptable trade-off between gain in privacy vs. loss of utility, i.e., process discovery remains useful while the disclosure of sensitive data is reduced. The follow-up paper [42] focuses on technological privacy-preserving challenges by introducing differential privacy for process discovery. The authors approach the problem by defining a protection model for event log privacy. They introduce a privacy engine that acts as a single point of access for process mining algorithms and introduces noise to each query result from log files in order to maintain differential privacy guarantees. There are two interesting aspects that distinguish their approach from our setup: first, they operate on typical process mining input, i.e., log lines with activity names and timestamps, and existing trace identifiers. In our solution, we record activities without a predefined case identifier. Second, additional attributes are usually restricted or completely ignored, while our solution preserves rich attribute information and linkage between objects.

Our approach requires privacy solutions at the intersection of process mining and graph anonymization. In the following, we discuss approaches to protect the privacy/confidentiality of process graphs:

Prune value nodes. This solution is specific to the *meta model* of our graph-structured TE𝔸M model. Since values are separated from the instance types, we have the possibility to remove value nodes while keeping the types and their relations for mining. While this approach offers an easy solution to remove identifiers and potential quasi-identifiers, mining algorithms cannot take advantage of the rich object information. It is also possible to remove only identified sensitive types or type relations, while preserving others.

Node replacement. A common technique to protect the identity of users inside a social graph is to remove sensitive information (e.g., ids, e-mail addresses, phone numbers, actual names) from nodes by replacing them with random identifiers. A similar approach can also be applied to the graph structures introduced in this paper by replacing the values corresponding to potentially sensitive object types with other identifiers (e.g., hashes or random values). Such replacements signify pseudonymization, which is a measure to make identification more difficult, but cannot be considered anonymous. The replacement can be one-way (e.g., deleting salts after hashing) or reversible, by keeping the information on how the data has been created (e.g., mapping lists).

Replacement methods to reach anonymity include generalization (e.g., replace date by year) and suppression (e.g., masking parts of the zip code).

Any solution implementing node replacement has to deal with the specifics of our data model, such as the uniqueness and re-use of value nodes. Replacement configurations could be defined once for the core model and reused by companies working with the TE𝔸M System. In case of company specific extensions and changes, the anonymization configuration would have to be adapted as well.

Graph anonymization. As mentioned in Section 5.1, the aforementioned approaches (pruning value nodes and node replacement) potentially leave the graph vulnerable to re-identification by taking advantage of the graph structure itself combined with background knowledge of the attacker. To give examples, despite removing all value nodes, it could still be possible to i) identify a company inside the graph, if we know that this company had contact with two different law firms recently, ii) identify employee nodes by knowing specific interactions (e.g., answered 5 calls from the same phone number in a short period of time), or iii) find the client node with most open patent cases.

Applying graph anonymization on our heterogeneous structures is not straight-forward and an open research challenge. While the basic ideas from e.g., [50, 56] could be adapted to our structures, the utility – privacy trade-off must be carefully studied.

6. Mining and Learning in the TE𝔸M Model

Based on the recording and managing of incoming and outgoing data for knowledge and communication tasks we are able to discover and analyse temporal and logical relations from the underlying graph-based representation by applying mining techniques.

Established process mining techniques basically require structured data in the form of "flat" (denormalised) models with a certain a-priori knowledge about process instances (i.e., a case ID), as described in due course in Section 6.1. In contrast, graph-based mining algorithms can directly work on the very flexible data model (cf. Section 4) and rely on defined instance data dependencies. These dependencies support the different working styles of users and express what can happen *logically*. This may differ from the

presentation of what exactly happened *one after the other*. However, by mining instance data dependencies (which can be done on the type graph) no knowledge about an explicit control flow or concrete process instances is needed. By counting all corresponding instances, probabilities can be calculated and associated with the mining result without having to know the individual instances. Suitable graphical representations of the found relationships already offer an added value for the strategic view of processes (i.e., in certain dashboards). They further form the basis for forecasting models for predicting the next possible process step and performance measures or risk indicators.

In this section, we discuss related process mining approaches and present our solution to mining and learning in a multi-dimensional knowledge/process graph. This novel, instance data dependency-based mining approach aims (i) to find typical patterns in the graph (knowledge and process discovery), (ii) to identify best practices for business process classes, such as registration of an intellectual property, and (iii) to suggest possible next tasks on these data to the user (prediction and enhancement).

6.1. Background

Process mining bridges the gap between traditional model-based process analysis in BPM (simulation, verification, optimization, etc.) and classical data analysis techniques (data mining, machine learning, etc.). It is a well-known technique for identifying, monitoring and improving business processes by extracting knowledge from process log data of information systems. Main characteristics of process mining are that it is not a specific type of data mining since it considers end-to-end processes and focuses on event data, requiring at least case ID, activity ID and the timestamp in the event log. The main difference to classical BPM is that BPM is based on a top-down approach (definition of the de jure process model), whereas process mining provides a bottom-up approach (identifying the de facto process model).

Mining business processes has become a major field of interest in recent years. In particular, existing work on process mining focuses on reconstructing meaningful process models from process instances [60, 61], mainly considering the control-flow perspective. A newer line of research concentrates on data-aware process mining [62, 63] to discover not only the control flow of a process but also the data flow and associated guards, which can then be added to a process model. Similarly, multi-perspective process explorer (MPE) [64] supports the discovery of data-aware process models based on data attributes attached to events. Further approaches on conformance checking [65, 66] also align an event log with data to evaluate the quality of discovered process models. They are typically based on Petri nets with data. A basic challenge, thereby, is that the discovered models tend to be complex and large, especially in flexible environments [67]. Thus, some approaches suggest the discovery of declarative process models [67, 68], where the discovered process behaviour is described as a set of rules.

All of these techniques to mine, analyse and check conformance of business processes have been developed based on event logs. This means that they require a case ID in the log, otherwise the techniques cannot be applied. If the case ID is missing, which is the case for both the administrative processes and highly flexible, knowledge-intense processes in the TEAM model, manual preprocessing of the log is necessary to assign a case ID to each event [69]. This might be done by correlating multiple IDs and dealing with many-to-many relationships (in case of cross-process relations), which is also referred to as "flattening reality" [61]. Nevertheless, a-priori knowledge of the process model

and driving key data objects is required. The issue of process mining without explicit case IDs is also addressed using sequence partitioning in [70, 71], which only works for simple workflow patterns, or for correlation mining in service-oriented systems [72]. The proposed approach for deducing case ids for unlabelled event logs in [69] requires an explicit process model, while the ICI approach for automated labelling of event log attributes [73] works without explicit process models but assumes that a case (trace) is explicitly mentioned in the log.

A brief survey on predictive monitoring of business processes is given in [74]. In [75] a predictive modelling approach, which also proposes a way to visualise the probabilistic process models (via a Petri net visualisation) is designed. However, this approach again only supports simple workflow patterns and requires a-priori knowledge of the process model. Further research on predicting process behaviour is given by early works using deep learning with recurrent neural networks (RNN) [76] and long short-term memory (LSTM) neural networks [77].

Since these approaches basically require structured data in the form of denormalized models or graph embeddings, the connection of such algorithms to the very flexible data storage, i.e., the dynamic, highly connected knowledge/process graph, is another major problem in addition to the missing case ID.

Relevant related work that deals with (instance) graphs as the underlying basis is given in [78, 79]. The authors propose a multi-step approach to aggregate a number of instance graphs (i.e., the representation of an execution of a business process for a single case) to obtain an overall model for an entire dataset. In contrast to this, in our work all execution paths are naturally supported and stored in the different layers of the TE$\mathbb{A}$M model, which forms the basis for our mining approach.

Since real-world processes are often unstructured and result in "spaghetti-like" models, a flexible approach for Fuzzy Mining, i.e., adaptively simplifying mined process models to provide meaningful abstractions of operational processes, is described in[80]. We refer to some of the proposed transformation methods to remove edges and less significant nodes in the resulting graph mining results. Similarly, the work on graph summarization to reduce data complexity and on local pattern mining to identify interesting graph patterns and sequential structures presented in [81] is relevant to our approach.

6.2. Data Dependency-based Mining

The essence of knowledge work is that processes are not known a priori, but are often designed task by task on the basis of the available data and information. For this reason, we choose a data-driven approach to support knowledge work, without the need to define pre-known processes. In doing so, we also avoid the delimitation and identification of individual processes, because such delimitation is not possible in practice or only possible for sub-processes. For example, in the case of a new property right application, bibliographic data of the applicant will be reused if the applicant is already an existing client.

6.2.1. Model Elements for Data Dependency-based Mining

Due to the uniqueness of data objects in the graph, it can be ensured that data objects created, validated or invalidated by a task are linked to the first task as well as to the subsequent task when data objects are reused. The data objects are linked to the tasks

via task relations, directed from the first task to the subsequent one, as shown in Fig. 7. A task is linked to one or more subsequent tasks via one or more data objects. These data objects are created in the first task and are therefore linked to it via a task relation y_c of the kind *creates*. When used in subsequent tasks, these data objects are referenced by a task relation y_r of the kind *references* directed towards each of the subsequent tasks. Since several data objects are usually exchanged between two subsequent tasks, for the sake of clarity, we reduce all intermediate task relations and data objects to a instance data dependency represented by a single relation θ.

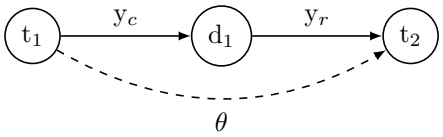


Figure 7: Instance data dependency θ (dashed) between two tasks t_1 and t_2 linked by a common data object d_1.

6.2.2. Data Dependency Paths

Under the condition of continuous instance data dependencies, for a given start and end node, all data dependency paths between both nodes describe the data flow between start and end node. The relevant data dependency paths can, for example, be found by executing a k-shortest path mining algorithm based on the start and end nodes. It makes no difference whether start and end nodes are tasks or data objects, but since a data object can only be created in a task, we assume one start task and one end task in the following.

6.2.3. Issues of the Data-driven Approach

In contrast to control flow-based mining, the problem with a data-driven approach is that there are not only instance data dependencies between two consecutive tasks, but that instance data dependencies may or may not exist to multiple tasks independently of their control flow. The resulting problems are illustrated in Fig. 8.

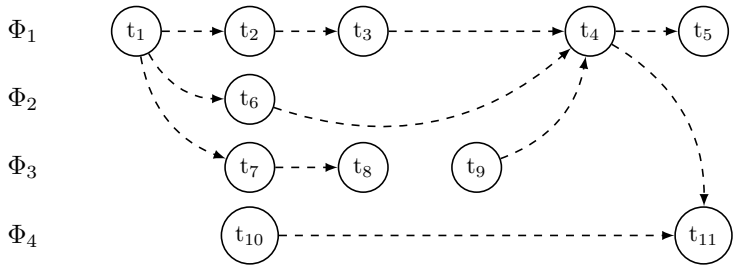


Figure 8: Identifying different data dependency paths Φ_1 - Φ_4 between a start node t_1 and an end node t_{11}.

Graph mining from start node t_1 to destination node t_{11} poses a number of structural problems.

Mining the longest path. Only two complete data dependency paths can be identified between t_1 and t_{11}, namely Φ_1 and Φ_2. Although Φ_2 is the shortest data dependency path, it hides the longer data dependency path Φ_1, especially the tasks t_2 and t_3.

From the perspective of mining, it is therefore, not the shortest data dependency path between start and end node that is relevant, but rather the longest data dependency path, which equally represents the critical process path from a data perspective.

Gaps within the data flow. However, if there is a gap within the data flow, as in data dependency path Φ_3, a potentially longer data dependency path might not be identified as there is no linking data object between intermediate tasks t_8 and t_9. As a consequence, tasks t_8 and t_9 would not be considered during graph mining.

Such a situation occurs, for example, when a data object leaves a system instance (e.g., sending an e-mail), tasks are performed outside the system instance, and then a resulting data object (e.g., a response e-mail) is reentered into the system instance through a task (e.g., receive an e-mail) without establishing a proper task relation to the last preceding task within the system instance.

In a complex, integrated knowledge/process graph it is unlikely that not a single data object is exchanged between two actually consecutive tasks, but if this is actually the case, it is possible to infer the actual sequence of tasks over the runtimes ζ_t of the tasks and the time ζ_θ elapsed between the end of the preceding and the beginning of the following task. Although this does not yet allow to reconstruct any missing instance data dependency, the tasks can be aligned to a time grid, which, analogous to classic process mining, does indicate the order in which the tasks are performed.

Connected partial paths. There are usually side paths from the longest path between start and destination node. For the data flow from t_1 to t_{11} the data provided by t_4 for t_5 is irrelevant. Data created by t_{10} however, may be a precondition for t_{11}.

Therefore, the search for all connecting paths between a start and an end node has to be supplemented with connected tasks and/or data objects starting from each task of the connecting paths found.

Explicitly and implicitly related data objects. Unique data objects are necessary to build reliable instance data dependencies. To reduce the complexity of the system for users, they rather use explicit references to usually more complex data objects in a task.

A large part of especially subordinate data objects, such as cities, postal codes, countries, etc. are only referenced implicitly, for example by assigning them to a new address data object within the respective task. These implicitly referenced data objects not only make up a large part of the resulting instance data dependencies, but - at least compared to classical process mining - hardly contribute to the progress of the local process.

Due to the lack of a clear tree structure, this problem also occurs in the selection of data objects that should be displayed to a user, wherefore a separate task relation of kind *displays* was introduced, giving an indication on which data objects are actually relevant to a user in the context of a task.

In the context of graph mining, we refer to these task relations to identify (i) explicit instance data dependencies, which contain data objects that are displayed by both interconnected tasks, and (ii) implicit instance data dependencies, which lack common displayed data objects.

Mining boundaries. One of the main issues with instance data dependency-based mining on instance level of the proposed model is the strong interconnection between tasks across process boundaries. This problem increases with growing graph size, because with regard to data objects an increasing saturation occurs, while mainly data object relations between already existing data objects are created or modified.

Although suggestions for future tasks could be made on the basis of task sequences carried out so far, it is not possible to identify typical task sequences and distinguish them from untypical ones in a meaningful way.

6.2.4. Data Dependency-based Mining on Type Level

Therefore, our approach follows mining at type level, because this allows insights about statistically relevant task sequences even without identifng individual processes. For this purpose, we introduce a type data dependency Θ, analogous to the instance data dependency θ, to which we assign properties aggregated from the instance model.

Transition probability and other measures. On type level, apart from simple item counts, a key metric is the transition probability for task type relations υ, i.e., the probability P_υ that, given a start node (task or data object) of a certain start node type $\upsilon.from$, a certain number of task relations exists, that link said starting node with a subsequent end node (data object or task) of a certain end node type $\upsilon.to$. For type data dependency, we calculate a similar transition metric for the probability P_Θ, that given a start task of a certain task type $\Theta.from$, a certain number of instance data dependencies exists, that link said starting task with a subsequent task of task type $\Theta.to$.

In addition, we aggregate an average execution time ζ_ξ of tasks and apply the results to the relevant task types. Together with an average transition period ζ_Θ assigned to type data dependencies, all task types can be again aligned to a common time grid.

Possible suggestions based on the mining approach. Although the described approach may not yet deliver satisfactory results with regard to long, continuous task chains, users can still be supported with reliable suggestions for action, especially in the local environment of some task nodes. For example, possible follow-up task types can first be selected on the basis of the type data dependency transition probabilities, then ranked in ascending order according to the average length of the time intervals, and finally those task types can be removed, that still have unfulfilled instance data dependencies. The proposed task types can be further limited by allowing the user to define data objects, data object types or task types that should be part of his further steps.

Advantages and disadvantages of type-based mining. The main advantage of this approach is that the introduction of case IDs is basically obsolete, even though the expected mining results are not directly comparable with classical process mining. Due to the fact that the type model depicts all possible type data dependencies and not only those which are actually manifested in an instance data dependency, data dependency-based mining on type level creates an options space.

If, for example, two process types A and B were recorded in the instance model, where A comprises an end task of a task type, that corresponds to the task type of the initial task of B, then a coherent data dependency path results in the type model, which has no equivalent in the instance model.

The significance of the mining results is therefore particularly dependent on the granularity of the assigned types. The more specific the assigned task types (and data object types as well) are, the more valid the data flow can be mapped and more specific type data dependencies are identified.

The alignment of the task types in a time grid further depends on the consistency of the time intervals at instance level. If the time intervals diverge too much, a sensible temporal alignment of the task types can no longer be done globally, which is why contradictions may arise between different type data dependencies with regard to the actual timing of the task types. In addition, time alignment is by its very nature extremely sensitive to cyclical data dependency paths, but this can be largely avoided by introducing specific types as described above.

Nevertheless, the proposed approach allows for proposals that are essentially based only on local properties of the graph (i.e., local transition probability and mean time intervals) and are therefore also applicable to very large knowledge/process graphs.

6.3. Test Data Generation

In order to validate our mining approach we were inspired by the methods in [82, 83] and developed our own synthetic event log generation tool. Existing tools, such as PLG [84] and Gena [85] address the problem of missing real-word execution logs and offer generators to simulate process models and capture the generated event logs. However, they typically focus on standard event logs without simulating the resources interacting with the running processes, which is a fundamental aspect in our approach. Hence, we developed our own log generator in order to have a valid ground truth of a trademark application scenario, which follows two stages as described in [83]: in the first stage *Model Generation* domain experts define processes in a chosen modelling notation, such as BPMN. In the second stage *Log Generation*, event logs are generated based on the previously defined process models.

As input our solution accepts a BPMN 2.0 XML model with custom statements encoded in *bpmn:documentation* elements. For objects, which are the main connectors in our model, we use Data Object References, which are either created in process activities or reused with a certain probability between process runs (e.g., the same client can file different trademark applications and the same lawyers work on multiple processes inside the law firm). Each Task takes 0 to n Data Objects as input and can create or modify 0 to n Data Objects as output. Furthermore, each Task defines an execution duration range (e.g., [30, 60] minutes) from which a random duration value will be drawn in each simulation run. For Gateways we define a transition probability for each outgoing Sequence Flow. Finally, the Start Event contains a starting time range, from which a date is randomly drawn for each process run.

Fig. 9 shows an example trademark application process[7]. In our Java generator, we take the exported BPMN 2.0 XML model as input and generate event logs for n simulated process runs in XML. The XML events are then imported into our process graph database for analysis.

[7]Modeled with `https://cawemo.com`

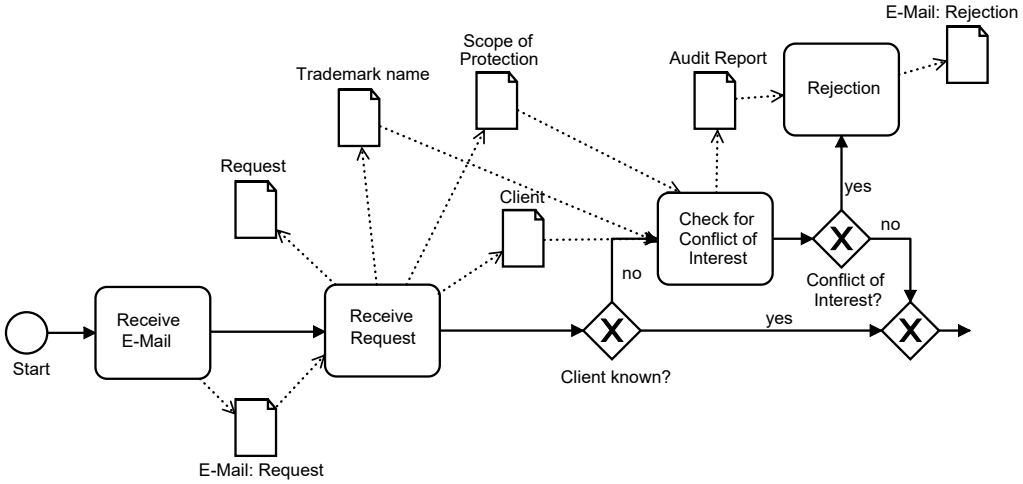


Figure 9: Trademark application process model (excerpt)

6.4. Determining the Ground Truth using Traditional Process Mining

Traditional process mining is used to analyse historical event data regarding the tasks of the communication processes from a process perspective. We, thereby, made use of the generated case IDs of the different process runs from our generator tool to be able to apply established process mining algorithms.

We first created the event log by querying the graph database and extracted all recorded tasks (instances). Since all relevant information (administrative and knowledge tasks) is available within one single TE𝔸M instance, we did not have to face issues such as incomplete or incorrect cases. We considered the task names as activities to determine the steps in the process as well as their start and end dates as timestamps to determine the temporal order in the process. The case IDs determine the scope of the process.

Fig. 12a illustrates the result (directly follows graph) of a standard process mining tool (ProM 6.9) used on the synthetic event logs with case IDs. This representation was used to verify the correct generation of process runs and is considered as ground truth for our instance data dependency-based mining approach, which does not use any case IDs .

7. Results

In this section, we sum-up the most important findings from answering the three research questions given in Section 2.

The results from developing the TE𝔸M model to answer RQ1 are summarized in Section 7.1, while Section 7.2 provides the answers to RQ2 dealing with privacy in the context of the TE𝔸M model. Finally, Section 7.3 shows details concerning mining task sequences in graphs and predicting best next steps in order to answer RQ3.

7.1. TE𝔸M Model (RQ1)

To answer the research question RQ1 concerning the support of highly accurate knowledge representation for data-driven, highly adaptable and individual processes, we developed the TE𝔸M model introduced in Section 4.

The meta model in the three-layer TE𝔸M model is the stable part of the overall model. It consists of a conceptual (type model) and an instance level (instance model) which are related to each other via the `type` relation. The type model defines all type level aspects such as (i) the sort of concepts (i.e., data object type (δ) and data object type relation (ρ) for the data view, and task type (ξ), including the integrating task type relation (υ) for the task view), and (ii) how to specify them, e.g., by name, creation date, etc. Thus, it is the type model in the meta model which defines how to describe our domain specific ontologies. The meta model further defines how to describe the instances.

The high level of flexibility and adaptability of the TE𝔸M model is achieved via the data-centric domain model and instance model. These two models rely on the stable meta model and can both be continuously adjusted and extended also at runtime.

To support data-driven processes, data quality and especially the constraint that instances are unique are very important. Further, the data interfaces of tasks are defined by their incoming and outgoing data objects.

Initial rather technical tests, analytical proofs of concepts including discussions and feedback circles with external experts, but also the case study shows that the TE𝔸M model provides the anticipated means to accurately define the concepts and instances for integrated highly-adaptable but also predefined processes. Furthermore, the information tracked while working on the system, promises to be an adequate source to generate additional knowledge which can be used to improve user support. The practical value of this approach for the legal domain has been demonstrated in a prototype, which was provided to five patent law firms in Austria. In the context of a case study, actual case data from existing management systems was imported and user feedback was continuously collected from administrative staff (8 users) and knowledge workers (9 users). Details on the user interface prototype as well as the insights gained are documented in [86].

7.2. Privacy (RQ2)

The overall research question RQ2 in regards to privacy was how and to what extent we can protect confidential information in the process graph management and mining. In Section 5.1, we discussed current results for enabling privacy-preserving, graph-based process sharing and mining.

Due to the domain model carrying semantic information on the connections between the various type nodes, we can apply a value pruning approach. This deletes actual values (such as names, dates, and textual data), however, while still keeping the semantic description inside the graph. The mining approach presented in this paper can still produce conforming process models from this structure alone. In this settings, most of the inference issues are reduced.

Node replacement can implement a form of pseudonymization, where we do not delete all values, but replace identifying attributes (such as names) with pseudonyms. This enables interlinking different processes along, e.g., common actors, w/o revealing their true identity, which is helpful in further analysis of the processes. It, however, introduces a larger attack surface for an adversary trying to infer information from the data.

Adapting k-anonymity for the value nodes that contain information on quasi-identifiers, such as dates or location information, is another option. This significantly reduces the attack surface for re-identification attacks. In future work, we will design a setting on which also differential privacy can be used for value nodes. This requires well-defined query interfaces and use-cases.

Furthermore, the graph structure could enable certain inference attacks, hence, graph anonymization, i.e., a deliberate blurring of the connections in the graph, needs to be incorporated into our solution.

We can measure the extent of privacy-preservation on the one hand by measures such as the achieved level of k for k-anonymity, or achieved values of ϵ in a Differential Privacy system. However, these remain rather abstract measures. More practical measures include a comparison of the success rates of attacks to the confidentiality of the data in the process graph. For example, we can measure the re-identification possibilities by an attacker with certain background knowledge on a real-world dataset in a lab setting. Measuring these before and after applying privacy-preserving measures, we can determine the increase in privacy. These will then need to be compared to the loss in conformance of the mined process model, against one that was obtained from original, unabridged data.

7.3. Process and Data Dependency-based Mining (RQ3)

Finally, we address RQ3 and go for answering how and to what extent statistical, process and graph mining methods can support the discovery of data-dependent sequences and prediction of tasks in the TE$\mathbb{A}$M model. In Section 6, we presented the results that enable process and data dependency-based mining in our multi-dimensional knowledge/process graph.

According to Section 6.3, test data for 100 process instances has been imported into a TE$\mathbb{A}$M System to develop a mining approach for reconstructing the logical (i.e., data-dependent) sequences of tasks. We evaluated the resulting models in several iterations using visual inspections (e.g., to identify missing tasks, exceptional tasks or exceptional sequences) and improve our mining approach according to the gained insights.

The initial import results in 14097 data objects, 1680 tasks and 51959 task relations. Executing the proposed instance data dependency-based mining approach reduced these task relations to 4761 instance data dependencies, which were mapped to 95 type data dependencies. Fig. 10 shows the complete mining result, starting from start task type *Receive Request* to end task type *Invoice Certificate of Registration* without any filtering applied. Note that the figure aims to illustrate the structural complexity of data dependencies and, thus, the contained text is not primarily intended to be readable.

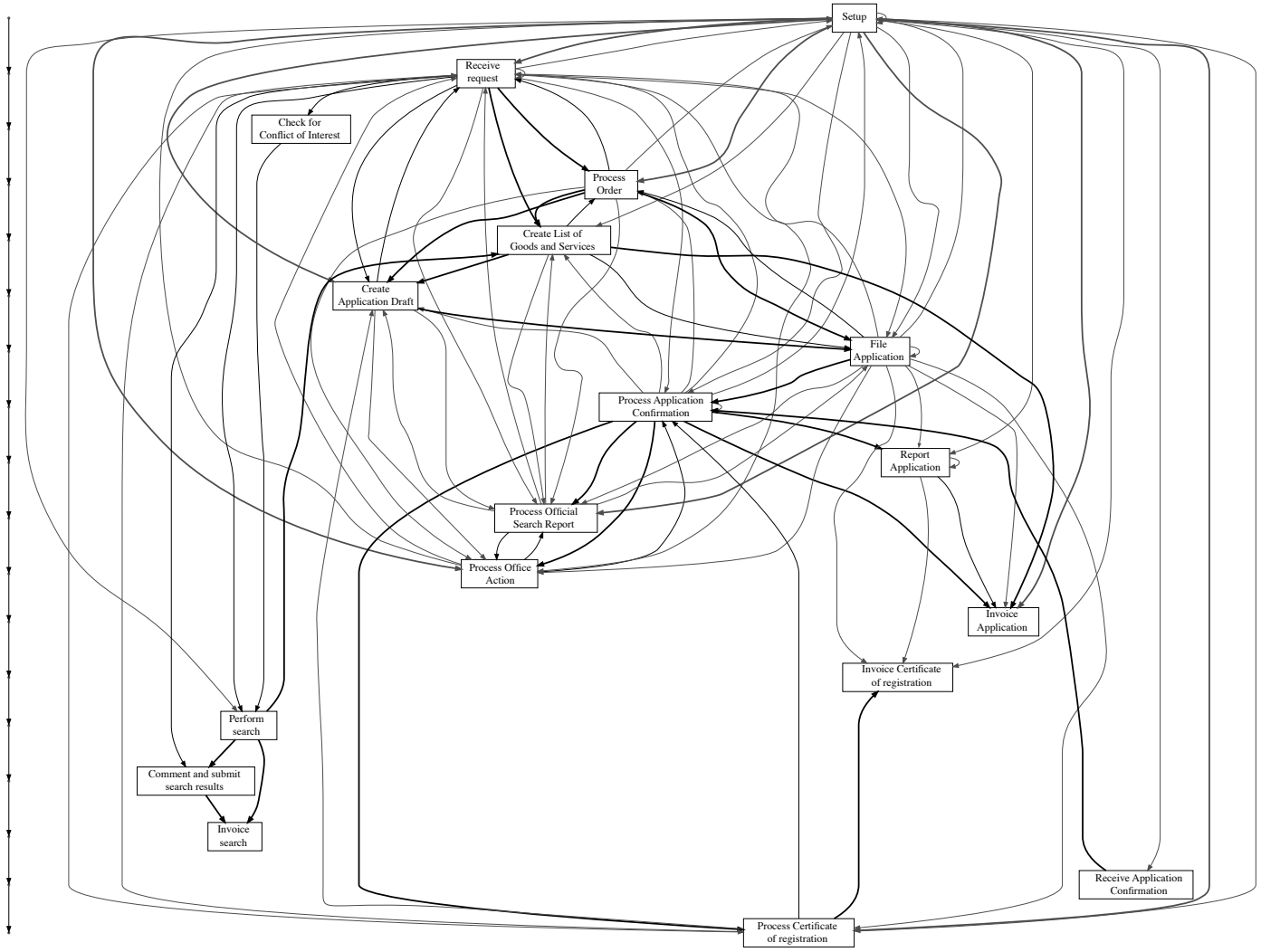


Figure 10: Visualised graph mining result with start task type *Receive Request* and end task type *Invoice Certificate of Registration*, showing type data dependencies with a probability of 100 % in bold.

The result shows that even with a visual simplification by representing type data dependencies as edges (i.e., by hiding explicit task type relations and data object types), a complex type data dependency graph emerges, which without further measures cannot easily be compared to the results of classic process mining, even if type data dependencies with a probability of 100 % are considered (shown in bold lines). The consideration of the typical running and transition times (indicated with time steps on the left) does not bring any recognizable added value either.

Particularly noticeable is the diversity of the occurring type data dependencies and that they not only connect successive task types, but often entire chains of successive task types (divergence). A further complicating factor is that there are also type data dependencies which at the first glance are not at all reflected in the modeled process. This is especially true for the circular self-references of some task types. The reason for these unexpected type data dependencies is that apparently single data objects are created in a task of a certain task type in one process instance and are referenced in another process instance by a task of the same task type (convergence). These cross-references between process instances also lead to cyclic data dependency paths, which hinder a correct temporal alignment of all task types considered.

However, such data dependency paths are rare and, therefore, not statistically relevant. If, for example, the type data dependencies are only restricted to a minimum probability of 5 %, as shown in Fig. 11, a large part of the type data dependencies are already omitted and a better temporal alignment is possible.

Considering type data dependencies with a probability of 100 %, the main sequence of task types begins to reveal. But still, several type data dependencies are not relevant for the progression of the task types sequence. In order to further improve the mining result in this respect, only those type data dependencies were considered in a further step, which were explicitly specified by the user. As shown in Fig. 12b, this can be used to eliminate additional type data dependencies that are not relevant for the alignment of the task types.

Referring to RQ3, the final mining result in Fig. 12b is comparable to the result of classical process mining shown in Fig. 12a. The research question is closely related to the practical challenge of discovering appropriate process models and delivering valuable artifacts for the application domain. The classical process mining result indicates a rigid task sequence. However, the identified type data dependencies indicate that there are stronger causal relationships between some task types, while instances of other task types can apparently run independently or at least parallel to each other. For example, loops make it difficult to grasp the main sequence of tasks in the process. The proposed data-driven mining approach better resolves the concurrences in the lower part of the process. Conversely, missing data dependencies (e.g., between the task types *Invoice search* and *Process Order*) lead to a task type order which can only be resolved by taking into account the time sequences. Furthermore, the task type *Receive Application Confirmation*, which is still present in Fig. 10, got omitted due to incomplete backward mining of type data dependencies for each of the resulting task type.

However, our approach allows for suggesting possible follow-up task types given a certain start node, without having to apply complex and resource-intensive graph mining techniques on the instance level. For example and with reference to Fig. 12b, starting from task type *Receive request*, a user could be supported by the suggestion of task type *Check for Conflict of Intereset*, as this is the temporally closest task type with a direct

type data dependency and no further required type data dependency. Alternatives are the task types *Perform Search* and *Process Order*; the task types *Comment and submit search results* and *Create List of Goods and Services*, however, are ruled out due to additional required type data dependencies with a transition probability of 100 %.

In summary, the mining result in Fig. 12b is (i) less complex (i.e., measuring the complexity of a process model using the number of arcs, the overall complexity reduction is 11 %) and (ii) provides more appropriate information for understanding the logical structure of the process than the model derived from classical process mining in Fig. 12a, which only reflects the temporal ordering.

The result, thus, shows that the presented approach is promising, but that further adjustments to the mining algorithm are necessary in order to reliably identify, especially longer, data dependency paths despite of missing type data dependencies and to handle divergence and convergence issues [87, 88]. Relevant data object types for the process under analysis need to be selected and task types should be clustered according to these data object types. This makes it possible to further simplify the model and to obtain different but connected views of the process with respect to specific interests.

8. Conclusion

In the legal domain, the requirement of process traceability, learning from earlier process executions, and repetitive, reliable administrative processes collides with the need for creative, data-driven knowledge work under an ever changing legal framework.

Our contribution is an approach that goes beyond previous methods and integrates administrative work and creative work in a single graph-based model, which enables continuously traceable processes via interconnected tasks and data objects. Due to a flexible and adaptable type model, our approach allows for the evolution of a-priori unknown or unidentifiable processes, including initially unknown task types and data object types.

The TEAM model consists of a stable meta model which defines the core structure of the overall model, i.e., an instance model related to its classifying domain model by a type relation, and how to specify these two models. The domain model provides the domain-specific mental models (types) and their relations. This graph-structured model can be extended and adapted by users whenever necessary without changing the meta model or recompiling parts of the software. The instance model, which relies on the domain model, holds all entities (data, users and tasks), information and communication flows, execution-specific data and the dynamic assignment of tasks.

As we follow a data-driven approach, sequences of data object / task pairs are created when using this model for supporting work, documenting the data-dependencies of each task. With our data dependency-based mining approach on type level, neither case IDs nor mining the instances is needed, as we focus on statistically relevant task sequences. Furthermore, the results are not restricted to a set of process executions, but define an options space, which contains all possible execution paths. A drawback of the approach is that long data dependency paths can only be analyzed if a fine-grained type model is available. However, the results for near neighboring nodes are reliable, and suggestions for the next steps can be very well supported.

In our future work, we will focus on the automated refinement of the type model based on historical data and, furthermore, work on improvements of the mining algorithms to

identify gaps and deal with dead ends. Further testing is also necessary. First, we will study additional kinds of processes within our testbed to study the influence of cross-process links. Based on the results of this study, we will undertake another test run in a real-world environment.

With process mining often sensitive data are involved. Therefore, privacy remains a difficult challenge in process mining, especially when considering collaborative settings, where different organizations need to exchange data and information. In this paper, we highlighted a number of selected approaches to enable privacy-preserving process graph sharing and mining. Future work will focus on measuring success rates of attacks against k-anonymization and graph anonymization solutions on the TE𝔸M model. Furthermore, we want to develop solutions to explore the trade-off between privacy and utility.

Acknowledgements

This work was funded within the FFG BRIDGE project KnoP-2D (grant no. 871299).

The work was also supported by the Austrian Ministry for Transport, Innovation and Technology, the Federal Ministry of Science, Research and Economy, and the Province of Upper Austria/federal state of Vienna in the frame of the COMET centers SCCH and SBA Research (SBA-K1) as well as by the LIT Secure and Correct Systems Lab funded by the State of Upper Austria.

The prototype development and testing in real-world settings was supported by polymind GmbH, Vienna, Austria.

References

[1] M. Dumas, M. La Rosa, J. Mendling, H. A. Reijers, Fundamentals of Business Process Management, Springer, 2018.

[2] M. Weske, Business process management: Concepts, languages, architectures, 2nd Edition, Springer, 2012.

[3] P. F. Drucker, Landmarks of Tomorrow: A Report on the New 'Post-Modern' World, Harper & Brothers, New York, 1959.

[4] P. F. Drucker, Knowledge-worker productivity: The biggest challenge, California Management Review Vol. 41 (No. 2) (1999) 79–94.

[5] D. E. Bailey, P. M. Leonardi, J. Chong, Minding the gaps: Understanding technology interdependence and coordination in knowledge work, Organization Science 21 (3) (2010) 713–730.

[6] D. Auer, S. Hinterholzer, J. Kubovy, J. Küng, Business process management for knowledge work: Considerations on current needs, basic concepts and models, in: F. Piazolo, M. Felderer (Eds.), Novel Methods and Technologies for Enterprise Information Systems, Vol. 8 of Lecture Notes in Information Systems and Organisation, Springer International Publishing, 2014, pp. 79–95.

[7] I. Nonaka, H. Takeuchi, The knowledge-creating company: How Japanese companies create the dynamics of innovation, Oxford university press, 1995.

[8] C. Di Ciccio, A. Marrella, A. Russo, Knowledge-intensive processes: Characteristics, requirements and analysis of contemporary approaches, Journal on Data Semantics 4 (1) (2015) 29–57.

[9] M. Marin, R. Hull, R. Vaculin, Data centric BPM and the emerging case management standard: a short survey, in: M. La Rosa, P. Soffer (Eds.), BPM 2012 Workshops, LNBIP, Springer, 2013, pp. 24–30.

[10] I. Bider, E. Perjons, Z. Riaz Dar, Using data-centric business process modeling for discovering requirements for business process support systems: Experience report, in: S. Nurcan, H. A. Proper, P. Soffer, J. Krogstie, R. Schmidt, T. Halpin, I. Bider (Eds.), Enterprise, Business-Process and Information Systems Modeling, LNBIP, Springer, 2013, pp. 63–77.

[11] A. R. Hevner, S. T. March, J. Park, S. Ram, Design science in information systems research, MIS Q 28 (1) (2004) 75–105.

[12] J. vom Brocke, A. R. Hevner, A. Maedche, Introduction to design science research, in: J. vom Brocke, A. Havner, A. Maedche (Eds.), Design science research, Progress in IS, Springer, 2020, pp. 1–13.
[13] F. Kossak, C. Illibauer, V. Geist, J. Kubovy, C. Natschläger, T. Ziebermayr, T. Kopetzky, B. Freudenthaler, K.-D. Schewe, A rigorous semantics for BPMN 2.0 process diagrams, Springer, 2014.
[14] H. Trætteberg, UI design without a task modeling language – using BPMN and Diamodl for task modeling and dialog design, in: P. Forbrig, F. Paternò (Eds.), Engineering Interactive Systems 2008, Vol. 5247 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2008, pp. 110–117.
[15] D. Auer, V. Geist, D. Draheim, Extending BPMN with submit/response-style user interaction modeling, in: 2009 IEEE Conference on Commerce and Enterprise Computing, IEEE, 2009, pp. 368–374.
[16] O. Marjanovic, Towards IS supported coordination in emergent business processes, Business Process Management Journal 11 (5) (2005) 476–487.
[17] M. Reichert, B. Weber, Enabling Flexibility in Process-Aware Information Systems: Challenges, Methods, Technologies, Springer, 2012.
[18] M. Döhring, B. Zimmermann, L. Karg, Flexible workflows at design- and runtime using BPMN2 adaptation patterns, in: W. Abramowicz (Ed.), Business Information Systems, Vol. 87 of Lecture notes in business information processing, Springer Berlin Heidelberg, 2011, pp. 25–36.
[19] W. M. van der Aalst, M. Weske, D. Grünbauer, Case handling: A new paradigm for business process support, in: Data and Knowledge Engineering, Elsevier B.V., 2005, pp. 129–162.
[20] OMG, Case Management Model and Notation: Version 1.0, OMG, 2014.
URL `http://www.omg.org/spec/CMMN/1.0`
[21] V. Künzle, Object-aware process management, Dissertation, Universität Ulm, Ulm (2013).
[22] S. Rinderle-Ma, S. Sadiq, F. Leymann, V. Künzle, M. Reichert, PHILharmonicFlows: towards a framework for object-aware process management, Journal of Software Maintenance and Evolution: Research and Practice 23 (4) (2011) 205–244.
[23] M. Pesic, W. M. van der Aalst, A declarative approach for flexible business processes management, in: D. Hutchison, et al. (Eds.), Business Process Management Workshops, Vol. 4103 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2006, pp. 169–180.
[24] A. Jiménez-Ramírez, B. Weber, I. Barba, C. Del Valle, Generating optimized configurable business process models in scenarios subject to uncertainty, Information and Software Technology 57 (2015) 571–594.
[25] C. Atkinson, D. Draheim, V. Geist, Typed business process specification, in: 2010 14th IEEE International Enterprise Distributed Object Computing Conference, IEEE, 2010, pp. 69–78.
[26] D. Draheim, G. Weber, Modeling submit/response style systems with form charts and dialogue constraints, in: OTM Confederated International Conferences "On the Move to Meaningful Internet Systems", Springer, 2003, pp. 267–278.
[27] D. Draheim, G. Weber, Form-oriented analysis: a new methodology to model form-based applications, Springer Science & Business Media, 2005.
[28] D. Draheim, Business process technology: A unified view on business processes, workflows and enterprise applications, Springer Science & Business Media, 2010.
[29] F. Kossak, C. Illibauer, V. Geist, C. Natschläger, T. Ziebermayr, B. Freudenthaler, T. Kopetzky, K.-D. Schewe, Hagenberg business process modelling method, Springer, 2016.
[30] K.-D. Schewe, B. Thalheim, Design and development of Web information systems, Springer, 2019.
[31] M. Frické, The knowledge pyramid: a critique of the DIKW hierarchy, Journal of information science 35 (2) (2009) 131–142.
[32] M. Hepp, F. Leymann, J. Domingue, A. Wahler, D. Fensel, Semantic business process management: a vision towards using semantic web services for business process management, in: IEEE International Conference on e-Business Engineering (ICEBE'05), IEEE, 2005, pp. 535–540.
[33] M. Hepp, D. Roman, An ontology framework for semantic business process management, in: Wirtschaftsinformatik Proceedings, 2007, pp. 423–440.
[34] J. Pujara, H. Miao, L. Getoor, W. Cohen, Knowledge graph identification, in: D. Hutchison, et al. (Eds.), Advanced Information Systems Engineering, Vol. 7908 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2013, pp. 542–557.
[35] H. Paulheim, Knowledge graph refinement: A survey of approaches and evaluation methods, Semantic Web 8 (3) (2016) 489–508.
[36] F. J. Ekaputra, Ontology-based data integration and knowledge change management in multi-disciplinary engineering environments, Dissertation, TU Wien, Vienna (2018).

[37] D. Saffer, Designing for interaction: Creating innovative applications and devices, 2nd Edition, Voices that matter, New Riders, Berkeley, CA, 2010.
[38] S. Nadschläger, J. Küng, A pattern collection for knowledge processing system architecture, in: V.-P. Eloranta, C. Preschern (Eds.), Proceedings of the 21st European Conference on Pattern Languages of Programs - EuroPlop '16, ACM Press, 2016, pp. 1–23.
[39] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, M. Stal, Pattern-Oriented Software Architecture, A System of Patterns, Wiley Software Patterns Series, Wiley, 2013.
[40] R. T. Fielding, Architectural styles and architectural styles and the design of network-based software architectures, Dissertation, University of California, Irvine (2000).
[41] A. Deepak, J. Crupi, D. Malks, Core J2ee Patterns: Best Practices and Design Strategies, Prentice Hall Computer, 2003.
[42] F. Mannhardt, A. Koschmider, N. Baracaldo, M. Weidlich, J. Michael, Privacy-preserving process mining, Business & Information Systems Engineering 61 (5) (2019) 595–614.
[43] L. Sweeney, K-anonymity: A model for protecting privacy, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 10 (5) (2002) 557–570.
[44] C. Dwork, Differential privacy, in: Proceedings of the 33rd International Colloquium on Automata, Languages and Programming ICALP, Vol. 4052 of Lecture Notes in Computer Science, Springer, 2006, pp. 1–12.
[45] B.-C. Chen, D. Kifer, K. LeFevre, A. Machanavajjhala, Privacy-preserving data publishing, Foundations and Trends in Databases 2 (1-2) (2009) 1–167.
[46] B. C. M. Fung, K. Wang, R. Chen, P. S. Yu, Privacy-preserving data publishing: A survey of recent developments, ACM Computing Surveys 42 (4) (2010) 14:1–14:53.
[47] L. Backstrom, C. Dwork, J. Kleinberg, Wherefore art thou r3579x? anonymized social networks, hidden patterns, and structural steganography, in: Proceedings of the 16th International Conference on World Wide Web, WWW '07, Association for Computing Machinery, 2007, p. 181–190.
[48] M. Hay, G. Miklau, D. Jensen, D. Towsley, P. Weis, Resisting structural re-identification in anonymized social networks, Proc. VLDB Endow. 1 (1) (2008) 102–114.
[49] B. Zhou, J. Pei, Preserving privacy in social networks against neighborhood attacks, in: 2008 IEEE 24th International Conference on Data Engineering, 2008, pp. 506–515.
[50] T. Feder, S. U. Nabar, E. Terzi, Anonymizing graphs (2008). arXiv:0810.5578.
[51] K. Liu, E. Terzi, Towards identity anonymization on graphs, in: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, SIGMOD '08, Association for Computing Machinery, 2008, p. 93–106.
[52] S. Bhagat, G. Cormode, B. Krishnamurthy, D. Srivastava, Class-based graph anonymization for social network data, Proc. VLDB Endow. 2 (1) (2009) 766–777.
[53] R. Mortazavi, S. Erfani, GRAM: An efficient (k, l) graph anonymization method, Expert Systems with Applications 153 (2020) 113454.
[54] C. C. Aggarwal, Y. Li, P. S. Yu, On the hardness of graph anonymization, in: 2011 IEEE 11th International Conference on Data Mining, 2011, pp. 1002–1007.
[55] S. Ji, P. Mittal, R. Beyah, Graph data anonymization, de-anonymization attacks, and de-anonymizability quantification: A survey, IEEE Communications Surveys Tutorials 19 (2) (2017) 1305–1326.
[56] A. Zhang, C. Gunter, X. Xie, J. Han, K. Chang, X. Wang, Privacy risk in anonymized heterogeneous information networks, in: V. Leroy, V. Christophides, V. Christophides, S. Idreos, A. Kementsietsidis, M. Garofalakis, S. Amer-Yahia (Eds.), Advances in Database Technology - EDBT 2014: 17th International Conference on Extending Database Technology, Proceedings, OpenProceedings.org, 2014, pp. 595–606.
[57] L.-E. Wang, X. Li, A graph-based multifold model for anonymizing data with attributes of multiple types, Computers & Security 72 (2018) 122–135.
[58] Y. A. A. S. Aldeen, M. Salleh, M. A. Razzaque, A comprehensive review on privacy preserving data mining, SpringerPlus 4 (1) (2015) 694.
[59] F. Mannhardt, S. A. Petersen, M. F. Oliveira, Privacy challenges for process mining in human-centered industrial environments, in: 2018 14th International Conference on Intelligent Environments (IE), 2018, pp. 64–71.
[60] W. Van Der Aalst, A. Adriansyah, A. K. A. De Medeiros, F. Arcieri, T. Baier, T. Blickle, J. C. Bose, P. Van Den Brand, R. Brandtjen, J. Buijs, et al., Process mining manifesto, in: International Conference on Business Process Management, Springer, 2011, pp. 169–194.
[61] W. Van Der Aalst, Data science in action, in: Process mining, Springer, 2016, pp. 3–23.
[62] M. De Leoni, W. M. van der Aalst, Data-aware process mining: discovering decisions in processes

using alignments, in: Proceedings of the 28th annual ACM Symposium on Applied Computing, 2013, pp. 1454–1461.
[63] F. Mannhardt, M. de Leoni, H. A. Reijers, W. M. van der Aalst, Data-driven process discovery-revealing conditional infrequent behavior from event logs, in: International Conference on Advanced Information Systems Engineering, Springer, 2017, pp. 545–560.
[64] F. Mannhardt, M. De Leoni, H. A. Reijers, The multi-perspective process explorer, BPM (Demos) 1418 (2015) 130–134.
[65] M. de Leoni, J. Munoz-Gama, J. Carmona, W. M. van der Aalst, Decomposing alignment-based conformance checking of data-aware process models, in: OTM Confederated International Conferences "On the Move to Meaningful Internet Systems", Springer, 2014, pp. 3–20.
[66] F. Mannhardt, M. De Leoni, H. A. Reijers, W. M. van der Aalst, Balanced multi-perspective checking of process conformance, Computing 98 (4) (2016) 407–437.
[67] F. M. Maggi, A. J. Mooij, W. M. van der Aalst, User-guided discovery of declarative process models, in: 2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM), IEEE, 2011, pp. 192–199.
[68] F. M. Maggi, M. Dumas, L. García-Bañuelos, M. Montali, Discovering data-aware declarative process models from event logs, in: Business Process Management, Springer, 2013, pp. 81–96.
[69] D. Bayomie, I. M. Helal, A. Awad, E. Ezat, A. ElBastawissi, Deducing case IDs for unlabeled event logs, in: International Conference on Business Process Management, Springer, 2016, pp. 242–254.
[70] D. R. Ferreira, D. Gillblad, Discovering process models from unlabelled event logs, in: International Conference on Business Process Management, Springer, 2009, pp. 143–158.
[71] M. Walicki, D. R. Ferreira, Mining sequences for patterns with non-repeating symbols, in: IEEE Congress on Evolutionary Computation, IEEE, 2010, pp. 1–8.
[72] S. Pourmirza, R. Dijkman, P. Grefen, Correlation mining: mining process orchestrations without case identifiers, in: International Conference on Service-Oriented Computing, Springer, 2015, pp. 237–252.
[73] A. A. Andaloussi, A. Burattin, B. Weber, Toward an automated labeling of event log attributes, in: Enterprise, Business-Process and Information Systems Modeling, Springer, 2018, pp. 82–96.
[74] A. E. Márquez-Chamorro, M. Resinas, A. Ruiz-Cortes, Predictive monitoring of business processes: a survey, IEEE Transactions on Services Computing 11 (6) (2017) 962–977.
[75] D. Breuker, M. Matzner, P. Delfmann, J. Becker, Comprehensible predictive models for business processes., MIS Q. 40 (4) (2016) 1009–1034.
[76] J. Evermann, J.-R. Rehse, P. Fettke, Predicting process behaviour using deep learning, Decision Support Systems 100 (2017) 129–140.
[77] N. Tax, I. Verenich, M. La Rosa, M. Dumas, Predictive business process monitoring with LSTM neural networks, in: International Conference on Advanced Information Systems Engineering, Springer, 2017, pp. 477–492.
[78] B. F. van Dongen, W. M. Van der Aalst, Multi-phase process mining: Aggregating instance graphs into EPCs and Petri nets, in: PNCWB 2005 workshop, Citeseer, 2005, pp. 35–58.
[79] B. F. Van Dongen, W. M. Van der Aalst, Multi-phase process mining: Building instance graphs, in: International Conference on Conceptual Modeling, Springer, 2004, pp. 362–376.
[80] C. W. Günther, W. M. Van Der Aalst, Fuzzy mining–adaptive process simplification based on multi-perspective metrics, in: International Conference on Business Process Management, Springer, 2007, pp. 328–343.
[81] M. Atzmueller, S. Bloemheuvel, B. Kloepper, A framework for human-centered exploration of complex event log graphs, in: International Conference on Discovery Science, Springer, 2019, pp. 335–350.
[82] A. A. Mitsyuk, I. S. Shugurov, A. A. Kalenkova, W. M. van der Aalst, Generating event logs for high-level process models, Simulation Modelling Practice and Theory 74 (2017) 1 – 16.
[83] T. Jouck, B. Depaire, Generating artificial event logs with sufficient discriminatory power to compare process discovery techniques, CEUR Workshop Proceedings (2014).
[84] A. Burattin, PLG2: multiperspective process randomization with online and offline simulations, in: L. Azevedo, C. Cabanillas (Eds.), Proceedings of the BPM Demo Track 2016 co-located with the 14th International Conference on Business Process Management (BPM), Vol. 1789 of CEUR Workshop Proceedings, CEUR-WS.org, 2016, pp. 1–6.
[85] I. S. Shugurov, A. A. Mitsyuk, Generation of a set of event logs with noise, in: Proceedings of the 8th Spring/Summer Young Researchers Colloquium on Software Engineering (SYRCoSE 2014), 2014, pp. 88–95.
[86] G. Hübscher, V. Geist, D. Auer, N. Hübscher, J. Küng, Integration of knowledge and task manage-

ment in an evolving, communication-intensive environment, in: The 22nd International Conference on Information Integration and Web-based Applications & Services (iiWAS '20), ACM, 2020, pp. 407–416.

[87] W. M. van der Aalst, Object-centric process mining: Dealing with divergence and convergence in event data, in: International Conference on Software Engineering and Formal Methods, Springer, 2019, pp. 3–25.

[88] W. M. van der Aalst, A. Berti, Discovering object-centric Petri nets, Fundamenta Informaticae 175 (1-4) (2020) 1–40.

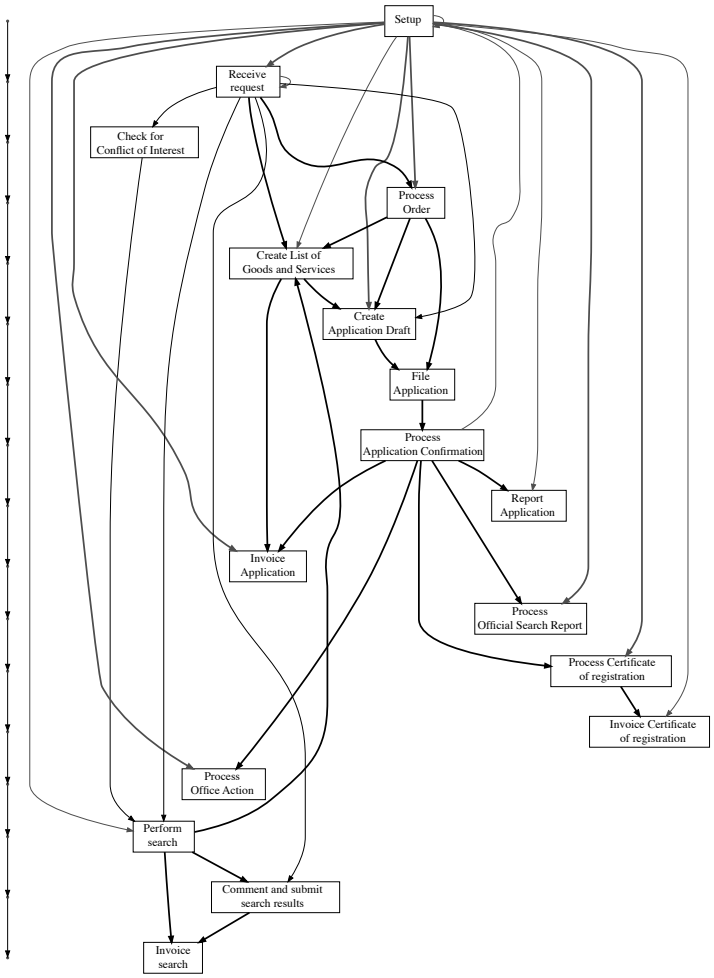
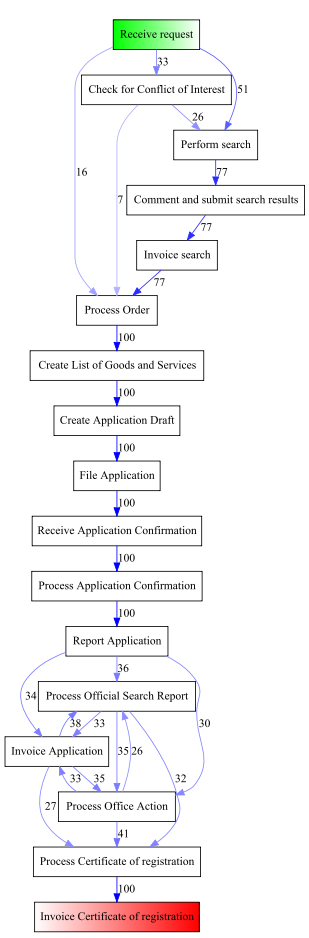
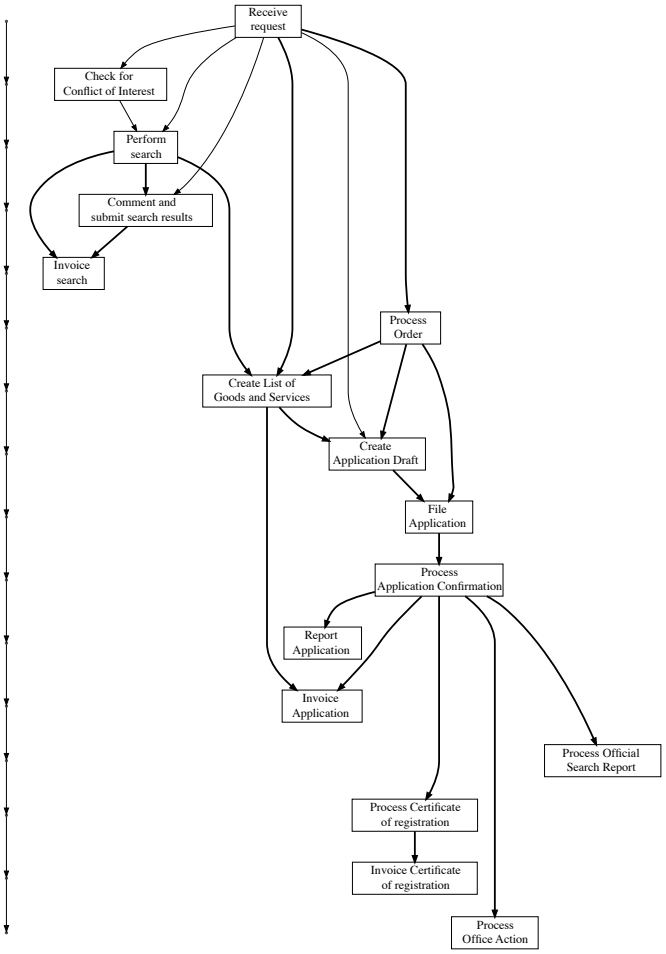


Figure 11: Visualised graph mining result according to Fig. 10, considering only type data dependencies with a probability of more than 5 %.



(a) Directly follows graph obtained from traces.



(b) Visualized graph mining result according to Fig. 10, considering only type data dependencies to data object types displayed by the user and at the same time having a probability of more than 5 %.

Figure 12: Final mining results