Artifact-Centric Process Mining (pre-print[1])

Dirk Fahland

Synonyms

Object-centric process mining, Multi-instance process mining

Definition

Artifact-centric process mining is an extension of classical process mining (van der Aalst 2016) that allows to analyze event data with more than one case identifier in its entirety. It allows to analyze the dynamic behavior of (business) processes that create, read, update, and delete multiple data objects that are related to each other in relationships with one-to-one, one-to-many, and many-to-many cardinalities. Such event data is typically stored in relational databases of, for example, Enterprise Resource Planning (ERP) systems (Lu et al 2015). Artifact-centric process mining comprises artifact-centric process discovery, conformance checking, and enhancement. The outcomes of artifact-centric process mining can be used for documentation of the actual data flow in an organization, and for analyzing deviations in the data flow for performance and conformance analysis.

The input to *artifact-centric process discovery* is either an event log where events carry information about the data objects and their changes, or a relational database also containing records about data creation, change, and deletion events. The output of artifact-centric process discovery are a data model of the objects (each defining its own case identifier) and relations between objects, and an artifact-centric pro-

Dirk Fahland
Eindhoven University of Technology, The Netherlands, e-mail: d.fahland@tue.nl

[1] This is the pre-print version of *Artifact-Centric Process Mining* in *Encyclopedia of Big Data Technologies*, *Editors: Sherif Sakr, Albert Zomaya*, Springer Cham, 2018. doi: http://dx.doi.org/10.1007/978-3-319-63962-8_93-1

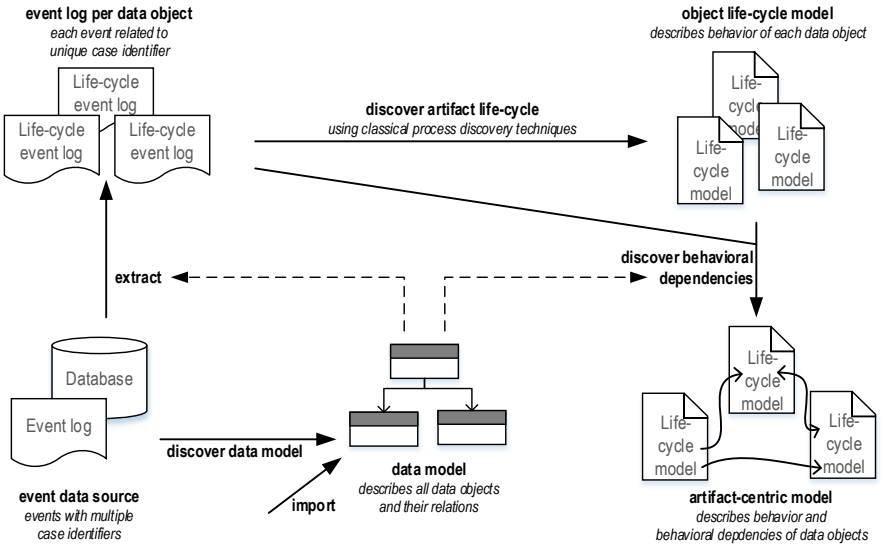

Fig. 1 Overview on artifact-centric process discovery.

cess model, as illustrated in Fig. 1. An artifact-centric process model describes the dynamics of each data object on its own in an *object life-cycle model*, and the *behavioral dependencies between the different data objects*. To this end, artifact-centric process discovery integrates the control-flow analysis of event data of classical process mining with an analysis of the data structures and data records related to the events.

During artifact-centric process discovery each event is associated to one data object in the data source. From the behavioral relations between all events associated to one data object, a *life-cycle model* of the data object is learned using automated process discovery techniques. Each life-cycle model describes the possible changes to the object and their ordering as they have been observed in reality. From behavioral relationships between events in different related data objects, information about *behavioral dependencies between changes in different data objects* is discovered preserving the one-to-one, one-to-many, and many-to-many cardinalities.

Several modeling languages have been proposed to describe a complete artifact-centric model of all object life-cycles and their behavioral interdependencies. Existing behavioral modeling languages can be extended to express interdependencies of one-to-many and many-to-many cardinalities including Petri nets (van der Aalst et al 2001) and UML (Estañol et al 2012). Specifically designed languages including the Guard-Stage-Milestone (GSM) model (Hull et al 2011) or Data-Centric Dynamic Systems (Hariri et al 2013) employ both data and behavioral constructs as primary modeling concepts. The Case Management Model and Notation (CMMN) standard v1.1 incorporates several modeling concepts of GSM (OMG 2016).

Artifact-centric conformance checking compares event data to an artifact-centric model with the aim to identify where recorded events *deviate* from the behavior described in the artifact-centric model. Deviations may exist between observed and specified data model, between observed events and the life-cycle model of an artifact, and between observed events and the interactions of two or more artifacts.

Artifact-centric model enhancement uses event data to enrich an artifact-centric model, for example with information about the frequency of paths through a life-cycle model or interactions, or to identify infrequent behavior as outliers.

Overview

Historically, artifact-centric process mining addressed the unsolved problem of process mining on event data with multiple case identifiers and one-to-many, and many-to-many relationships by adopting the concept of a (business) *artifact* as an alternative approach to describing business processes.

Event data with multiple case identifiers

Processes in organizations are typically supported by information systems to structure the information handled in these processes in well-defined data objects which are often stored in relational databases. During process execution, various data objects are created, read, updated, and deleted, and various data objects are related to each other in one-to-one, one-to-many, and many-to-many relations. Figure 2 illustrates in a simplified form the data structures typically found in ERP systems. *Sales*, *Delivery*, and *Billing* documents are recorded in tables; relation *F1* links *Sales* to *Delivery* documents in a one-to-many relation: *S1* relates to *D1* and *D2*, correspondingly *F2* links *Billing* to *Delivery* documents in a one-to-many relation. Events on process steps and data access are recorded in time-stamp attributes such as *Date created* or in a separate *Document Changes* table linked to all other tables.

Convergence and divergence

Process mining requires to associate events to a *case identifier* in order to analyze behavioral relations between events in the same case (van der Aalst 2016). The data in Fig. 2 provides *three* case identifiers: *SD id*, *DD id*, and *BD id*. Classical process mining forces to associate all events to a single case identifier. However, this is equivalent to flattening and de-normalizing the relational structure along its one-to-many relationships.

Sales Documents

SD id	Date created	Value	Last Change
S1	16-5-2020	100	10-6-2020
S2	17-5-2020	200	5-6-2020

Billing Documents

BD id	Date created	Clearing Date
B1	20-5-2020	31-5-2020
B2	24-5-2020	5-6-2020

F1 F2

Delivery Documents

DD id	Date created	Reference SD id	Reference BD id	Picking Date
D1	18-5-2020	S1	B1	31-5-2020
D2	22-5-2020	S1	B2	5-6-2020
D3	25-5-2020	S2	B2	5-6-2020

F3

Document Changes

Change id	Date	Ref. id	Table	Change type	Old Value	New Value
1	17-5-2020	S1	SD	Price updated	100	80
2	19-5-2020	S1	SD	Delivery block released	X	-
3	19-5-2020	S1	SD	Billing block released	X	-
4	10-6-2020	B1	BD	Invoice date updated	20-6-2020	21-6-2020

Fig. 2 Event data stored in a relational database.

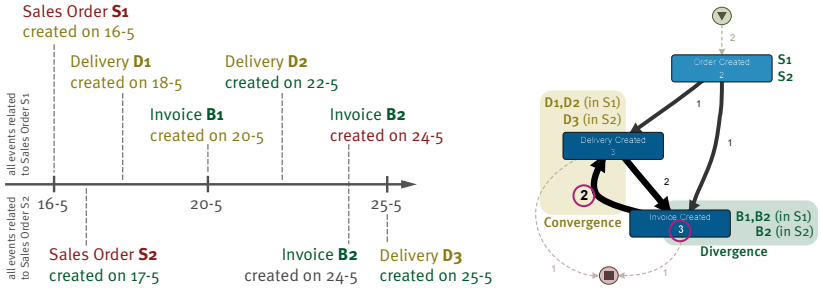

Fig. 3 An event log (left) serializing the "create" events of the database of Fig. 2 based on the case identifier "SD id". The resulting directly-follows relation (right) suffers convergence and divergence.

For example, associating all *Create* events of Fig. 2 to *SD id* flattens the tables into the *event log* of Fig. 3(left) having two cases for *S1* and *S2*. Due to flattening, event *Create* of *B2* has been duplicated as it was extracted once for *S1* and once for *S2*, also called *divergence*. Further, *Create* for *B1* is followed by *Create* for *D2* although *B1* and *D2* are unrelated in Fig. 2, also called *convergence* Lu et al (2015). The behavioral relations which underly automated process discovery become erroneous through convergence and divergence. For example, the directly-follows relation of the log (Fig. 3 right) states erroneously that 3 *Invoice* documents have been created – whereas the original data source contains only 2 – and that in 2 cases *Invoice* creation was followed by *Delivery* creation (between related data objects) – whereas in the original data source this only happened once for *B2* and *D3*. Convergence and divergence may cause up to 50% of erroneous behavioral relations (Lu et al 2015) in event logs. Convergence and divergence can be avoided partially by scoping extraction of event data into event logs with a single case identifier in a manual process (Jans 2017).

Artifact-centric process models

Artifact-centric process mining adopts modeling concept of a (business) artifact to analyze event data with multiple case identifiers in their entirety (Lu et al 2015; Nooijen et al 2012; van Eck et al 2017). The notion of a (business) *artifact* was proposed by Nigam and Caswell (2003) as an alternative approach to describing business processes. This approach assumes that any process materializes itself in the (data) objects that are involved in the process, for instance, sales documents and delivery documents; these objects have properties such as the values of the fields of a paper form, the processing state of an order, or the location of a package. Typically, a *data model* describes the (1) classes of objects that are relevant in the process, (2) the relevant properties of these objects in terms of class attributes, and (3) the relations between the classes. A process execution instantiates new objects and changes their properties according to the process logic. Thereby, the relations between classes describe how many objects of one class are related to how many objects of another class.

An *artifact-centric process model* enriches the classes of the data model themselves with process logic restricting how objects may evolve during execution. More precisely, one *artifact* (1) encapsulates several classes of the data model (e.g., *Sales Documents* and *Sales Document Lines*), (2) provides *actions* that can update the classes attributes and move the artifact to a particular state, and (3) defines a *life cycle*. The artifacts life cycle describes when an instance of the artifact (i.e., a concrete object) is created, in which state of the instance which actions may occur to advance the instance to another state (e.g., from *created* to *cleared*), and which goal state the instance has to reach to complete a case. A complete artifact-centric process model provides a life-cycle model for each artifact in the process and describes which *behavioral dependencies* exist between actions and states of different artifacts (e.g., *pick delivery* may occur for a *Delivery* object only if all its *Billing* objects are in state *cleared*). Where business process models created in languages such as BPMN, EPC, or Petri nets describe a process in terms of activities and their ordering in a single case, an artifact-centric model describes process behavior in terms of creation and evolution of instances of multiple related data objects. In an artifact-centric process model, the unit of modularization is the *artifact*, consisting of data and behavior, whereas in an activity-centric process modeling notation, the unit of modularization is the *activity*, which can be an elementary task or a sub-process. A separate entry in this encyclopedia discusses the problem of automated discovery of activity-centric process models with sub-processes.

Figure 4 shows an artifact-centric process model in the notation of *Proclets* (van der Aalst et al 2001) for the data model of Fig. 2. The life cycle of each document (*Sales*, *Delivery*, *Billing*) is described as a Petri net. Behavioral dependencies between actions in different objects are described through *interface ports* and *asynchronous channels* that also express cardinalities in the interaction. For example, the port annotation "+" specifies that *Clear Invoice* in *Billing* enables *Pick Delivery* in *multiple related Delivery* objects. Port annotation "1" specifies that *Pick Delivery* can occur after *Clear Invoice* occurred in the *one Billing* object related to the *Delivery*.

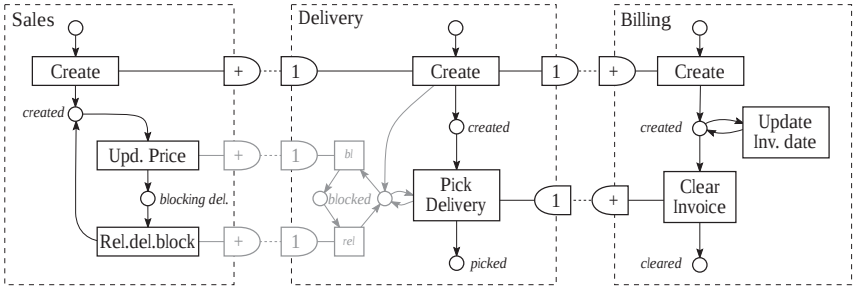

Fig. 4 Example of an artifact-centric process model in the Proclet notation.

The grey part of Fig. 4 shows a more involved behavioral dependency. Whenever *Update Price* occurs in a *Sales* document, all related *Delivery* documents get *blocked*. Only after *Release delivery block* occurred in *Sales*, the *Delivery* document may be updated again, and *Pick Delivery* may occur. For the sake of simplicity, the model does not show further behavioral dependencies such as "*Update price* also blocks related *Billing* documents".

Figure 5 shows the same model in the *Guard-Stage-Milestone* notation (Hull et al 2011) (omitting some details). Each round rectangles denotes a *stage* that can be entered when its *guard* condition (diamond) holds and is left when its *milestone* condition (circle) holds. The guard and milestone conditions specify declarative constraints over data attributes, stages, and milestones of *all* artifacts in the model. For example, *Picking* can start when the *pickDelivery* event is triggered in the process, the delivery document has reached its *created* milestone, the billing document related to the delivery document has reached its *cleared* milestone (*d.BD.cleared*), and stage *Blocking Delivery* is not active in the related sales document.

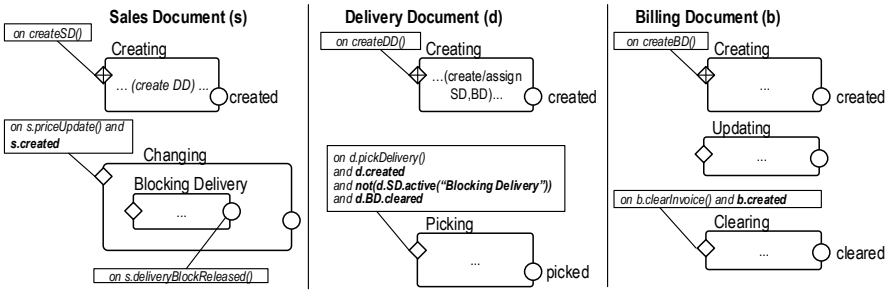

Fig. 5 Example of an artifact-centric process model in the Guard-Stage-Milestone notation.

Artifact-centric process mining

The behavior recorded in the database of Fig. 2 does not *conform* to the models in Figures 4 and 5.

1. *Structural conformance* states how well the data model describes the data records observed in reality. The proclet model of Fig. 4 structurally conforms to the data in Fig. 2 regarding objects and relations, but not regarding actions: the recorded event data shows two additional event types for the life cycle of the *Sales* document: *Release billing block* and *Last Change*.
2. *Life-cycle conformance* states how well the life-cycle model of each artifact describes the order of events observed in reality. This corresponds to conformance in classical process mining. For example, in Fig. 2, *Update invoice date* occurs in *Billing* after *Clear Invoice* which does not conform to the life-cycle model in Fig. 4.
3. *Interaction conformance* states how well the entire artifact centric model describes the behavioral dependencies between artifacts. In Fig. 2, instance *D3* of *Delivery* is *created* after its related instance *B2* of *Billing*. This does not conform to the channels and ports specified in Fig. 4.

The objective of artifact-centric process mining is to relate recorded behavior to modeled behavior, through (1) *discovering* an artifact-centric process model that conforms to the recorded behavior, (2) *checking* how well recorded behavior and an artifact-centric model *conform* to each other and detecting *deviations*, and (3) extending a given artifact-centric model with further information based on recorded event data.

Artifact-centric process discovery is a technique to automatically or semi-automatically learn artifact-centric process models from event data. The problem is typically solved by a decomposition into the following four steps:

1. Discovering the data model of entities or tables, their attributes, and relations from the data records in the source data. This step corresponds to *data schema recovery*. It can be omitted if the data schema is available and correct, however in practice foreign key relations may not be documented at the data level and need to be discovered.
2. Discovering artifact types and relations from the data model and the event data. This steps corresponds to transforming the data schema discovered in step 1 into a domain model often involving undoing horizontal and vertical (anti-) partitioning in the technical data schema and grouping entities into domain-level data objects. User input or detailed information about the domain model are usually required.
3. Discovering artifact life-cycle models for each artifact type discovered in step 2. This step corresponds to automated process discovery for event data with a single case identifier, and can be done fully automatically up to parameters of the discovery algorithm.
4. Discovering behavioral dependencies between the artifact life cycles discovered in step 3 based on the relations between artifact types discovered in step 2. This

step is specific to artifact-centric process mining; several alternative, automated techniques have been proposed. User input may be required to select domain-relevant behavioral dependencies among the discovered ones.

In case the original data source is a relational database, steps 3 and 4 require to *automatically extract event logs* from the data source for discovering life-cycle models and behavioral dependencies. As in classical process discovery, it depends on the use case to which degree the discovered data model, life-cycle model, and behavioral dependencies shall conform to the original data.

Artifact-centric conformance checking and *Artifact-centric model enhancement* follow the same problem decomposition into data schema, artifact types, life cycles, and interactions as artifact-centric discovery. Depending on which models are available, the techniques may also be combined by first discovering data schema and artifact types, then extracting event logs, and then checking life-cycle and behavioral conformance for existing model, or enhancing an existing artifact model with performance information.

Figure 6 shows a possible result of artifact-centric process discovery on the event data in Fig. 2 using the technique of Lu et al (2015) where the model has been enhanced with information about the *frequencies* of occurrences of events and behavioral dependencies.

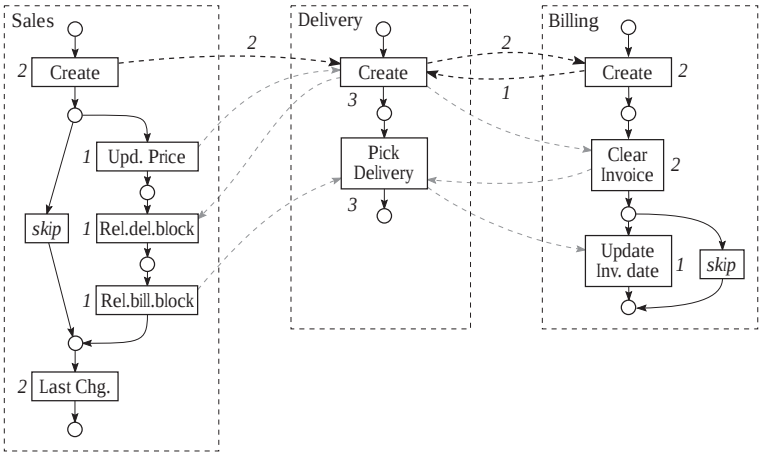

Fig. 6 Possible result of artifact-centric process discovery from the event data in Fig. 2

Key Research Findings

Artifact-type discovery. Nooijen et al (2012) provides a technique for automatically discovering artifact types from a relational database, leveraging schema summariza-

tion techniques to cluster tables into artifact types based on information entropy in a table and the strength of foreign key relations. The semi-automatic approach of Lu et al (2015) can then be used to refine artifact types and undo horizontal and vertical (anti-)partitioning, and to discover relations between artifacts. Popova et al (2015) shows how to discover artifact types from a rich event stream by grouping events based on common identifiers into entities, and then deriving structural relations between them.

Event log extraction. In addition to discovering artifact types, Nooijen et al (2012) also automatically creates a mapping from the relational database to the artifact type specification. The technique of Verbeek et al (2010) can use this mapping to generation queries for event log extraction for life-cycle discovery automatically. Jans (2017) provides guidelines for extracting specific event logs from databases through user-defined queries. The event log may also be extracted from database redo logs using the technique of de Murillas et al (2015), and from databases through a meta-model based approach as proposed by de Murillas et al (2016).

Life-cycle discovery. Given the event log of an artifact, artifact life-cycle discovery is a classical automated process discovery problem for which various process discovery algorithms are available, most returning models based on or similar to Petri nets. Weerdt et al (2012) compared various discovery algorithms using real-life event logs. Lu et al (2015) advocates the use of the Heuristics Miner of Weijters and Ribeiro (2011) and vanden Broucke and Weerdt (2017) with the aim of visual analytics. Popova et al (2015) advocates to discover models with precise semantics and free of behavioral anomalies, that (largely) fit the event log (Leemans et al 2013; Buijs et al 2012) allowing for translating the result to the Guard-Stage-Milestone notation.

Behavioral dependencies. Lu et al (2015) discovers behavioral dependencies between two artifacts by extracting an *interaction event log* that combines the events of any two related artifact instances into one trace. Applying process discovery on this interaction event log then allows to extract "flow edges" between activities of the different artifacts, also across one-to-many relations, leading to a model as shown in Fig. 6. This approach has been validated to return only those dependencies actually recorded in the event data, but suffers when interactions can occur in may different variants, leading to many different "flow edges".

van Eck et al (2017) generalizes the interaction event log further and creates an integrated event log of all artifact types to be considered (two or more) where for each combination of related artifact instances, all events are merged into a single trace. From this log a composite state machine model is discovered which describes the synchronization of all artifact types. By projecting the composite state machine onto the steps of each artifact type, the life-cycle model for each artifact is obtained and the interaction between multiple artifacts can be explored interactively in a graphical user interface through their relation in the composite state machine. This approach assumes one-to-one relations between artifacts.

Popova and Dumas (2013) discovers behavioral dependencies in the form of data conditions over data attributes and states of other artifacts, similar to the notation in Fig. 5 but is limited to one-to-one relations between artifacts.

Conformance checking. Artifact life-cycle conformance can be checked through extracting artifact life-cycle event logs and then applying classical conformance checking techniques Fahland et al (2011a). The technique in Fahland et al (2011b) checks interaction conformance in an artifact life-cycle model if detailed information about which artifact instances interact is recorded in the event data.

Models for artifacts. Artifact-centric process mining techniques originated and are to a large extent determined by the modeling concepts available to describe process behavior and data flow with multiple case identifiers. Several proposals have been made in this area. The proclet notation van der Aalst et al (2001) extended Petri nets with ports that specify one-to-many and many-to-many cardinality constraints on messages exchanged over channels in an asynchronous fashion. Fahland et al (2011c) discusses a normal form for proclet-based models akin to the second normal form in relational schemas. The Guard-Stage-Milestone (GSM) notation Hull et al (2011) proposes allows to specify artifacts and interactions using event-condition-actions rules over the data models of the different artifacts. Several modeling concepts of GSM were adopted by the CMMN 1.1 standard of OMG (2016). Hariri et al (2013) propose Data-Centric Dynamic Systems (DCDS) to specify artifact-centric behavior in terms of updates of data-base records using logical constraints. Existing industrial standards can also be extended to describe artifacts as shown by Lohmann and Nyolt (2011) for BPMN and by Estañol et al (2012) for UML. Freedom of behavioral anomalies can be verified for UML-based models (Calvanese et al 2014) and for DCDS (Montali and Calvanese 2016). Meyer and Weske (2013) show how to translate between artifact-centric and activity-centric process models, and Lohmann (2011) show how to derive an activity-centric process model describing the interactions between different artifacts based on behavioral constraints.

Examples of Application

Artifact-centric process mining is designed for analyzing event data where events can be related to more than one case identifier or object, and where more than one case identifer has to be considered in the analysis.

The primary use case is in analyzing processes in information systems storing multiple, related data objects, such as Enterprise Resource Planning (ERP) systems. These systems store documents about business transactions that are related to each other in one-to-many and many-to-many relations. Lu et al (2015) correctly distinguish normal and outlier flows between 18 different business objects over two months of data of the Order-to-Cash process in an SAP ERP system using artifact-centric process mining. The same technique was also used for identifying outlier behavior in processes of a project management system together with end users. van Eck et al (2017) analyzed the personal loan and overdraft process of a Dutch financial institution. Artifact-centric process mining has also been applied successfully on software project management systems such as Jira and customer relationship management systems such as Salesforce (Calvo 2017).

Artifact-centric process mining can also be applied on event data outside information systems. One general application area is analyzing the behavior of physical objects as sensed by multiple related sensors. For instance, van Eck et al (2016) analyzed the usage of physical objects equipped with multiple sensors. Another general application area is analyzing the behavior of software components from software execution event logs. For instance, Liu et al (2016) follow the artifact-centric paradigm to structure events of software execution logs into different components and discover behavioral models for each software component individually.

Future Directions for Research

At the current stage, artifact-centric process mining is still under development allowing for several directions for future research.

Automatically discovering artifact types from data sources is currently limited to summarizing the structures in the available data. Mapping these structure to domain concepts still requires user input. Also the automated extraction of event logs from the data source relies on the mapping from the data source to the artifact type definition. How to aid the user in discovering and mapping the data to domain-relevant structures and reducing the time and effort to extract event logs, possibly through the use of ontologies, is an open problem. Also little research has been done for improving the queries generated for automated event log extraction to handle large amount of event data.

For discovering behavioral dependencies between artifacts, only few and limited techniques are available. The flow-based discovery of (Lu et al 2015) that can handle one-to-many relations is limited to interactions between two artifacts and suffers in the presence of many different behavioral variants of the artifacts or the interactions. The alternative approaches (Popova and Dumas 2013; van Eck et al 2017) are currently limited to one-to-one relations between artifacts. Solving the discovery of behavioral dependencies between artifacts thereby faces two fundamental challenges.

1. Although many different modeling languages and concepts for describing artifact-centric processes have been proposed, the proposed concepts do not adequately capture these complex dynamics in an easy-to-understand form (Reijers et al 2015). Further research is needed to identify appropriate modeling concepts for artifact interactions.
2. Systems with multiple case identifiers are in their nature complex systems, where complex behaviors and multiple variants in the different artifacts multiply when considering artifact interactions. Further research is needed on how to handle this complexity, for example through generating specific, interactive views as proposed by van Eck et al (2017).

Although several, comprehensive conformance criteria in artifact-centric process mining have been identified, only behavioral conformance of artifact life-cycles can

currently be measured. Further research for measuring structural conformance and interaction conformance is required, not only for detecting deviations, but also to objectively evaluate the quality of artifact-centric process discovery algorithms.

Cross-References

Business process analytics, Automated process discovery, Hierarchical process discovery, Conformance checking, Multidimensional process analytics, Schema Mapping

References

van der Aalst WMP (2016) Process Mining - Data Science in Action, Second Edition. Springer, DOI 10.1007/978-3-662-49851-4, URL https://doi.org/10.1007/978-3-662-49851-4

van der Aalst WMP, Barthelmess P, Ellis CA, Wainer J (2001) Proclets: A framework for lightweight interacting workflow processes. Int J Cooperative Inf Syst 10:443–481

vanden Broucke SKLM, Weerdt JD (2017) Fodina: A robust and flexible heuristic process discovery technique. Decision Support Systems 100:109–118

Buijs JCAM, van Dongen BF, van der Aalst WMP (2012) A genetic algorithm for discovering process trees. In: IEEE Congress on Evolutionary Computation

Calvanese D, Montali M, Estañol M, Teniente E (2014) Verifiable uml artifact-centric business process models. In: CIKM

Calvo HAS (2017) Artifact-centric log extraction for cloud systems. Master's thesis, Eindhoven University of Technology, the Netherlands

van Eck ML, Sidorova N, van der Aalst WMP (2016) Composite state machine miner: Discovering and exploring multi-perspective processes. In: BPM

van Eck ML, Sidorova N, van der Aalst WMP (2017) Guided interaction exploration in artifact-centric process models. 2017 IEEE 19th Conference on Business Informatics (CBI) 01:109–118

Estañol M, Queralt A, Sancho MR, Teniente E (2012) Artifact-centric business process models in uml. In: Business Process Management Workshops

Fahland D, de Leoni M, van Dongen BF, van der Aalst WMP (2011a) Behavioral conformance of artifact-centric process models. In: BIS

Fahland D, de Leoni M, van Dongen BF, van der Aalst WMP (2011b) Conformance checking of interacting processes with overlapping instances. In: BPM

Fahland D, de Leoni M, van Dongen BF, van der Aalst WMP (2011c) Many-to-many: Some observations on interactions in artifact choreographies. In: ZEUS

Hariri BB, Calvanese D, Giacomo GD, Deutsch A, Montali M (2013) Verification of relational data-centric dynamic systems with external services. In: PODS

Hull R, Damaggio E, Masellis RD, Fournier F, Gupta M, Heath FT, Hobson S, Linehan MH, Maradugu S, Nigam A, Sukaviriya N, Vaculín R (2011) Business artifacts with guard-stage-milestone lifecycles: managing artifact interactions with conditions and events. In: DEBS

Jans M (2017) From Relational Database to Event Log: Decisions with Quality Impact. In: First International Workshop on Quality Data for Process Analytics

Leemans SJJ, Fahland D, van der Aalst WMP (2013) Discovering block-structured process models from event logs - a constructive approach. In: Petri Nets

Liu CS, van Dongen BF, Assy N, van der Aalst WMP (2016) Component behavior discovery from software execution data. 2016 IEEE Symposium Series on Computational Intelligence (SSCI) pp 1–8

Lohmann N (2011) Compliance by design for artifact-centric business processes. In: Inf. Syst.

Lohmann N, Nyolt M (2011) Artifact-centric modeling using bpmn. In: ICSOC Workshops

Lu X, Nagelkerke M, van de Wiel D, Fahland D (2015) Discovering interacting artifacts from erp systems. IEEE Transactions on Services Computing 8:861–873

Meyer A, Weske M (2013) Activity-centric and artifact-centric process model roundtrip. In: Business Process Management Workshops

Montali M, Calvanese D (2016) Soundness of data-aware, case-centric processes. International Journal on Software Tools for Technology Transfer 18:535–558

de Murillas EGL, van der Aalst WMP, Reijers HA (2015) Process mining on databases: Unearthing historical data from redo logs. In: BPM

de Murillas EGL, Reijers HA, van der Aalst WMP (2016) Connecting databases with process mining: A meta model and toolset. In: BMMDS/EMMSAD

Nigam A, Caswell NS (2003) Business artifacts: An approach to operational specification. IBM Systems Journal 42:428–445

Nooijen EHJ, van Dongen BF, Fahland D (2012) Automatic discovery of data-centric and artifact-centric processes. In: Business Process Management Workshops, Springer, Lecture Notes in Business Information Processing, vol 132, pp 316–327, DOI 10.1007/978-3-642-36285-9_36, URL https://doi.org/10.1007/978-3-642-36285-9_36

OMG (2016) Case Management Model and Notation, Version 1.1. URL http://www.omg.org/spec/CMMN/1.1

Popova V, Dumas M (2013) Discovering unbounded synchronization conditions in artifact-centric process models. In: Business Process Management Workshops

Popova V, Fahland D, Dumas M (2015) Artifact lifecycle discovery. Int J Cooperative Inf Syst 24

Reijers HA, Vanderfeesten ITP, Plomp MGA, Gorp PV, Fahland D, van der Crommert WLM, Garcia HDD (2015) Evaluating data-centric process approaches: Does the human factor factor in? Software & Systems Modeling 16:649–662

Verbeek HMW, Buijs JCAM, van Dongen BF, van der Aalst WMP (2010) Xes, xesame, and prom 6. In: CAiSE Forum

Weerdt JD, Backer MD, Vanthienen J, Baesens B (2012) A multi-dimensional quality assessment of state-of-the-art process discovery algorithms using real-life event logs. Inf Syst 37(7):654–676, DOI 10.1016/j.is.2012.02.004, URL https://doi.org/10.1016/j.is.2012.02.004

Weijters AJMM, Ribeiro JTS (2011) Flexible heuristics miner (fhm). 2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM) pp 310–317