Xixi Lu, Dirk Fahland : TU/e
Dennis van de Wiel, Marijn Nagelkerke : KPMG IT Advisory N.V., Eindhoven, The Netherlands

TU/e Technische Universiteit Eindhoven University of Technology

Process Mining for ERP Systems
The Artifact-Centric Approach

read more: http://dx.doi.org/10.1109/TSC.2015.2474358, contact: d.fahland@tue.nl

Event Data in a Relational Database

Enterprise Resource Planning (ERP) systems - and many other information systems - **record events** about changes to information such as creation or update of a business object.

Often, the objects are in complex **1-to-many or many-to-many relations** - and so are the recorded events.

The changes to business objects of the **Order to Cash (OTC) process** shown below are a typical example of event data embedded in a complex structure.

Sales documents (SD)

SD id	Date created	Reference id	Document type	Value	Last change
S1	16-5-2020	null	Sales Order	100	10-6-2020
S2	17-5-2020	null	Sales Order	200	31-5-2020
S3	10-6-2020	S1	Return Order	10	NULL

F1, F2

Delivery documents (DD)

DD id	Date created	Reference SD id	Reference BD	Document type	Picking date
D1	18-5-2020	S1	B1	Delivery	31-5-2020
D2	22-5-2020	S1	B2	Delivery	5-6-2020
D3	25-5-2020	S2	B2	Delivery	5-6-2020
D4	12-6-2020	S3	null	Return Delivery	NULL

F3

Billing documents (BD)

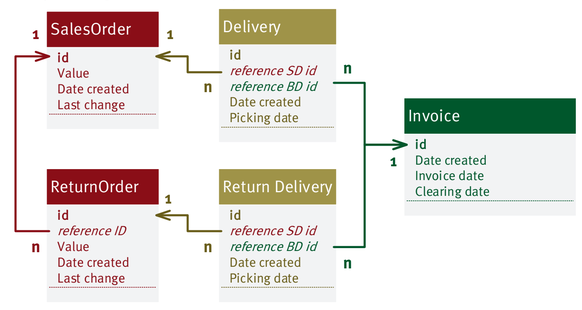
BD id	Date created	Document type	Clearing date
B1	20-5-2020	Invoice	31-5-2020
B2	24-5-2020	Invoice	5-6-2020

F4

Documents Changes

Change id	Date changed	Reference id	Table name	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

The underlying conceptual data model comprises 5 business objects.



The user may **refine the clusters**, for instance by allowing that several business objects share the same table.

Classical Log Extraction

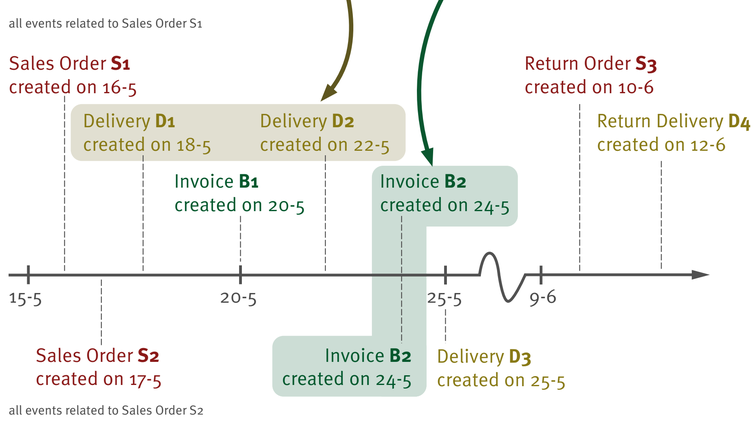
To analyze the events using process mining techniques, pick a **case identifier** for the process, for example the Sales Order id.

Any timestamp related to a value of the case identifier becomes an **event** of that case; sort events by their timestamp.

In case of many-to-many relations between events coming from different tables, two phenomena arise.

Divergence. Events of different instances of the same object are in the same case.

Convergence. Duplication of the same event into different cases.



Process Mining

A standard process mining technique then returns an end-to-end process across all the business objects touched by the process.

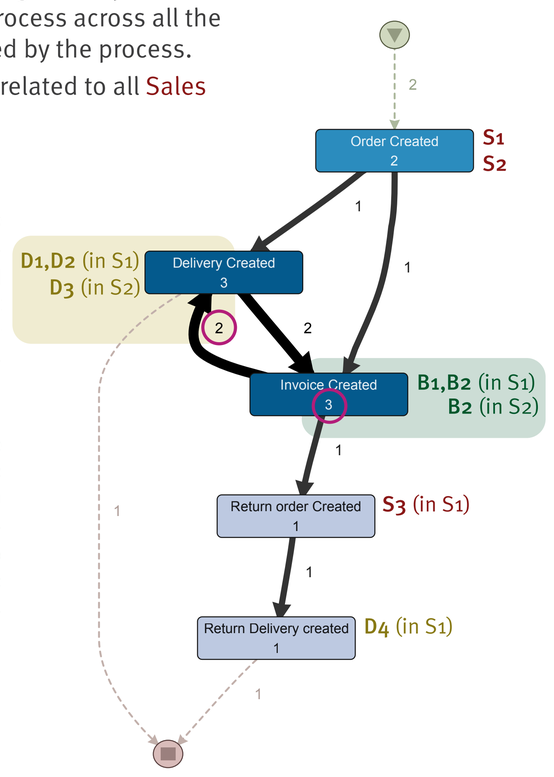
In this case the process related to all Sales Orders.

Divergence leads to **false edges** and skews statistics.

Only invoice B2 was created before its delivery D3, but the model suggest this happened twice.

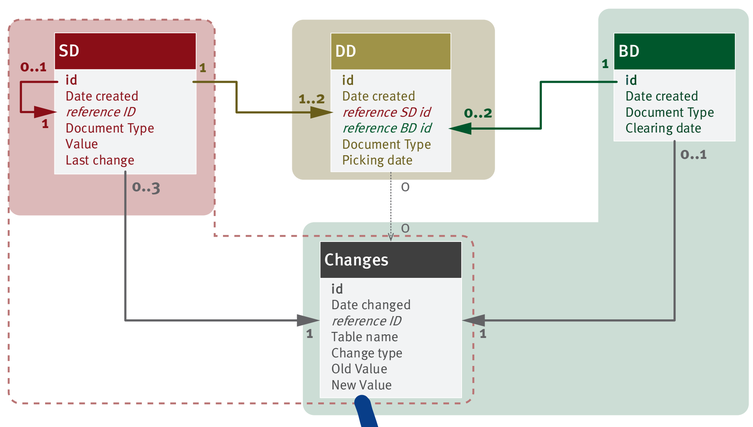
Convergence leads to **more events and flows** in the model than actually are in the data.

There have only been two invoices created, but the model shows three.



Extracting Artifact Types

We identify **clusters of connected tables** linked by **1-to-1 relations only**. Intuitively, each cluster describes **a schema of similar business objects**; within a business object, convergence and divergence cannot arise.



We split each schema of business objects into one or more **artifact types** describing **a specific business object.**

A **discriminating condition** distinguishes artifact types of the same schema.

The **primary key of the "main table"** of the schema becomes the **case id** of the artifact type.

Each **timestamp** attribute in the schema becomes an **event type** of the artifact.

Artifact Sales Order

Artifact Id	[SD id]
Condition	SD.[Document type] = 'Sales Order'
Event type	DateCreated
Event id	[SD id]
Timestamp	[date created]
Condition	
Event type	LastChange
Event id	[SD id]
Timestamp	[last change]
Condition	
Event type	PriceUpdated
Event id	[Change id]
Timestamp	[Date changed]
Condition	Changes.[Change type] = 'Price updated'
Event type	DeliveryBlockReleased
Event id	[Change id]
Timestamp	[Date changed]
Condition	Changes.[Change type] = 'Delivery block released'
Event type	BillingBlockReleased
Event id	[Change id]
Timestamp	[Date changed]
Condition	Changes.[Change type] = 'Billing block released'

Artifact Return Order

Artifact Id	[SD id]
Condition	SD.[Document type] = 'Return Order'
Event type	DateCreated
Event id	[SD id]
Timestamp	[date created]
Condition	

Artifact Return Delivery

Artifact Delivery

Artifact Id	[DD id]
Condition	SD.[Document type] = 'Delivery'
Event type	DateCreated
...	...

Artifact Invoice

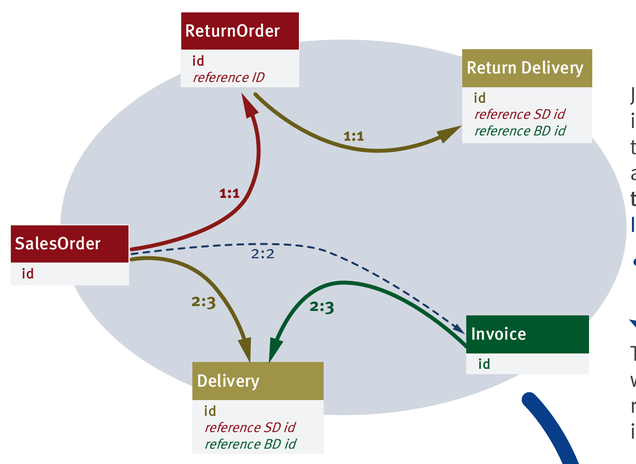
Artifact Id	[BD id]
Condition	
Event type	DateCreated
...	...

Extracting Artifact Interactions

The **1-to-many and many-to-many relations** between clusters and artifact types describe their **interactions**.

Each relation between to artifact types defines a **direct structural interaction** whihch we visualize in the **artifact interaction graph:**

3 instances of the Invoice artifact interact with 2 instances of the Delivery artifact.

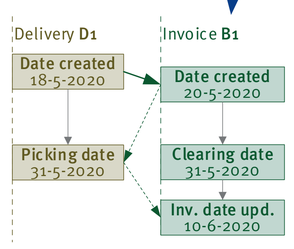


Joining two direct interactions at their target artifact yields an **indirect interaction** (Sales Order to Invoice).

The user can choose which direct or indirect interactions to include.

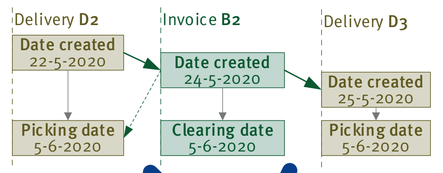
Each structural interaction between two artifacts defines the **artifact instances that interact** with each other.

To identify which events interact with each other, we **merge the event logs** of interacting instances.



Two **events interact** if they are in different artifacts and directly succeed each other timewise (other criteria can be defined)

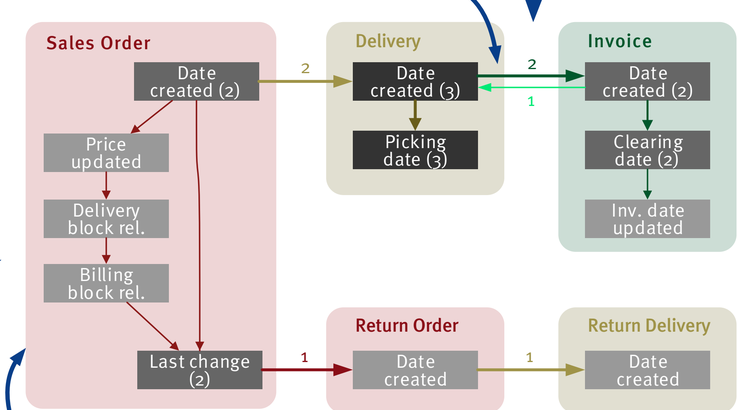
The user can filter which interactions shall be considered.



We add the aggregate interactions between events as **interaction flows** between activities of the artifact life-cyclde models

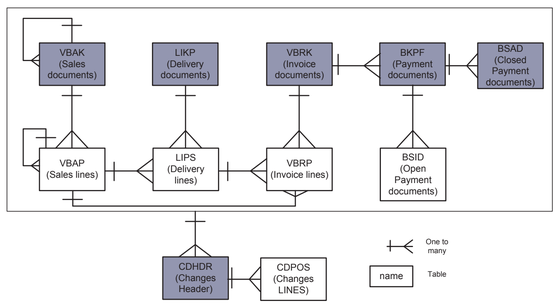
Date created of Delivery interacts with Date created of Invoice.

Interactions. Describe how different life-cycles synchronize; allows to study unusual flows between artifacts.



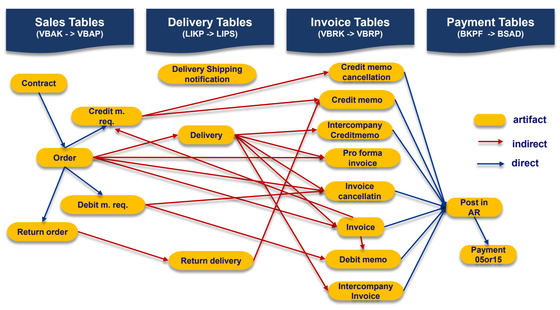
Life-cycle model. Describes how each instance of a business object evolves in the process; different instances are separated.

Analysis of SAP Order-to-Cash



We analyzed 2 months of data of an SAP Order-to-Cash process, focusing on all document headers.

We identified and extracted event data for 18 artifact types.

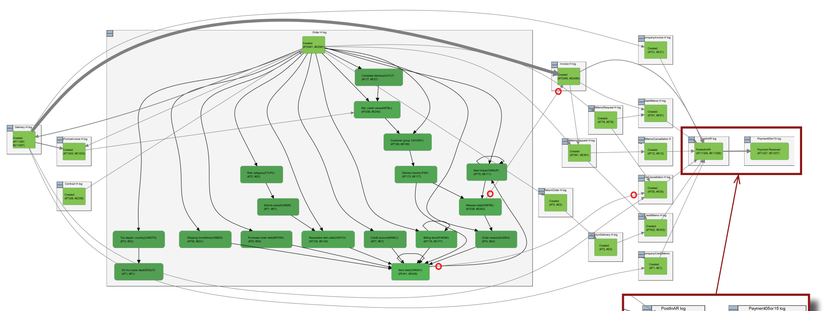


An artifact type may contain event types from all tables of the schema.

The user can **choose which event types to include.**

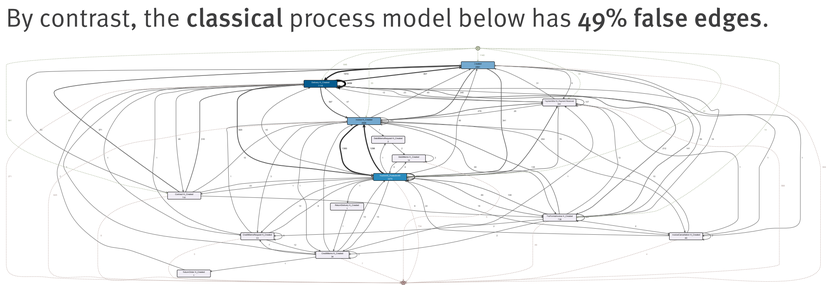
Event types can be **refined** by a discriminating condition.

The resulting artifact-centric model highlights the main flows across the different objects; object life-cycles can be **expanded** (here for the **Order** artifact) or **collapsed** to focus on specific aspects.



From the artifact-centric model various unusual flows can be identified automatically or by an analyst: **Payment050r15** has been executed too early in some cases.

By contrast, the **classical** process model below has **49% false edges**.



Mining Artifact Life-Cycles + Interactions

The artifact type definitions allow to fully **automatically extract** one **event log for each artifact type.** From this a **life-cycle model** for each artifact can be discovered with a classical process mining technique.