Getting the Data

Barbara Re

Process Mining

Data Sources

Data Sources

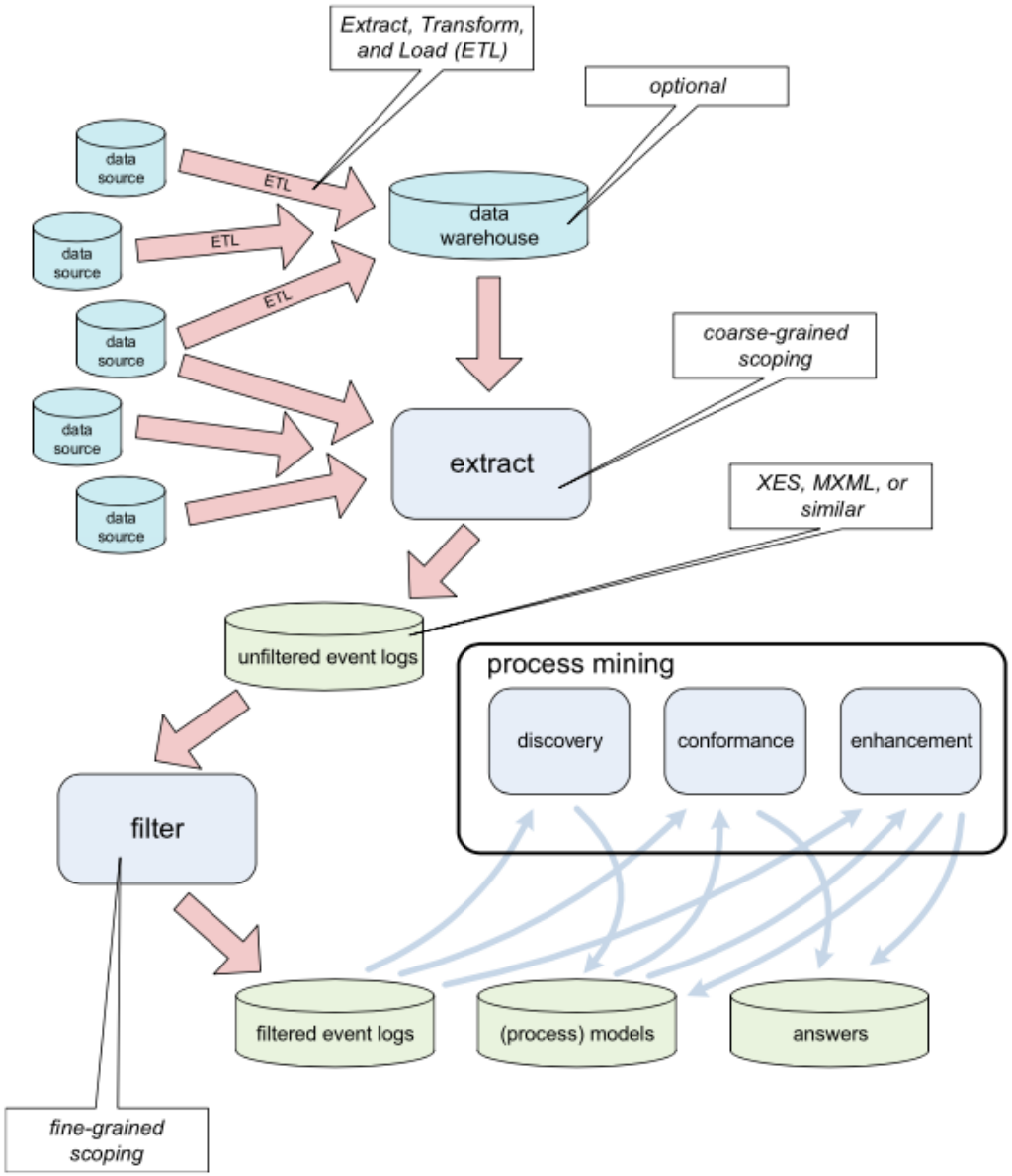
Process mining is impossible without **proper event logs**

- The challenge is to extract such data from a variety of data sources, e.g., databases, flat files, message logs, transaction logs, ERP systems, and document management systems

The goal of process mining is to answer questions about operational processes

- What really happened in the past?
- Why did it happen?
- What is likely to happen in the future?
- When and why do organizations and people deviate?
- How to control a process better?
- How to redesign a process to improve its performance?

Process Mining Workflow



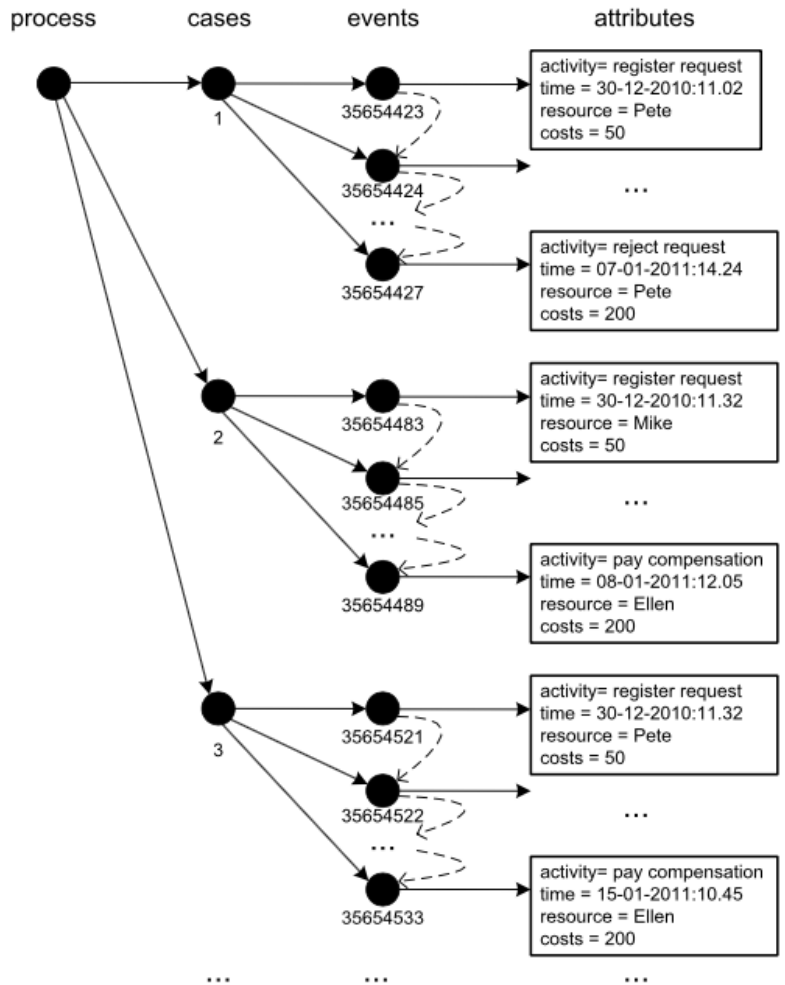
A fragment of some event log

Each line corresponds to an event

Case id	Event id	Properties				
		Timestamp	Activity	Resource	Cost	...
1	35654423	30-12-2010:11.02	register request	Pete	50	...
	35654424	31-12-2010:10.06	examine thoroughly	Sue	400	...
	35654425	05-01-2011:15.12	check ticket	Mike	100	...
	35654426	06-01-2011:11.18	decide	Sara	200	...
	35654427	07-01-2011:14.24	reject request	Pete	200	...
2	35654483	30-12-2010:11.32	register request	Mike	50	...
	35654485	30-12-2010:12.12	check ticket	Mike	100	...
	35654487	30-12-2010:14.16	examine casually	Pete	400	...
	35654488	05-01-2011:11.22	decide	Sara	200	...
	35654489	08-01-2011:12.05	pay compensation	Ellen	200	...
3	35654521	30-12-2010:14.32	register request	Pete	50	...
	35654522	30-12-2010:15.06	examine casually	Mike	400	...
	35654524	30-12-2010:16.34	check ticket	Ellen	100	...
	35654525	06-01-2011:09.18	decide	Sara	200	...
	35654526	06-01-2011:12.18	reinitiate request	Sara	200	...
	35654527	06-01-2011:13.06	examine thoroughly	Sean	400	...
	35654530	08-01-2011:11.43	check ticket	Pete	100	...
	35654531	09-01-2011:09.55	decide	Sara	200	...
	35654533	15-01-2011:10.45	pay compensation	Ellen	200	...
4	35654641	06-01-2011:15.02	register request	Pete	50	...
	35654643	07-01-2011:12.06	check ticket	Mike	100	...
	35654644	08-01-2011:14.43	examine thoroughly	Sean	400	...
	35654645	09-01-2011:12.02	decide	Sara	200	...
	35654647	12-01-2011:15.44	reject request	Ellen	200	...
...	...	...	...	...	...	...

Structure of event logs

- A process consists of cases (also called instance)
- A case consists of events such that each event relates to precisely one case
- Events within a case are ordered
- Events can have attributes (e.g. of typical attribute names are activity, time, costs, and resource)



Structure of event logs

Definition (Event)

Let $\mathcal{E}$ be the `event universe`, i.e., the set of all possible event identifiers

Events may be characterized by various `attributes`, e.g., an event may have a timestamp, correspond to an activity, is executed by a person, has associated costs, etc.

Definition (Attribute)

Let $\mathcal{AN}$ be a set of attribute names. For any `event` $e \in \mathcal{E}$ and `name` $n \in \mathcal{AN}$:

- $\#n(e)$ is the value of attribute n for event e
- If event e does not have an attribute named n, then $\#n(e) = \perp$ (null value)

Standard attributes we will assume some conventions

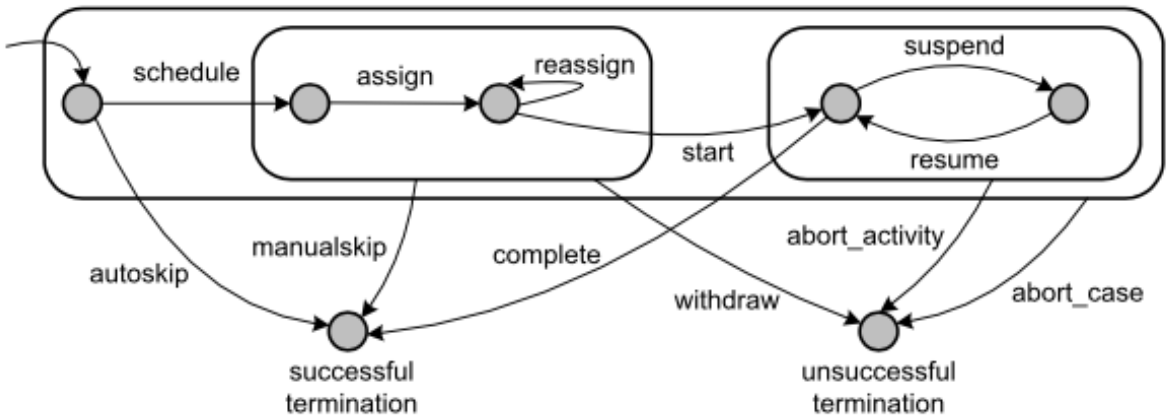
For convenience we assume the following standard attributes (none of these attributes is mandatory):

- $\#activity(e)$ is the activity associated to event e
- $\#time(e)$ is the timestamp of event e
- $\#resource(e)$ is the resource associated to event e
- $\#trans(e)$ is the transaction type associated to event e, examples are schedule, start, complete, and suspend

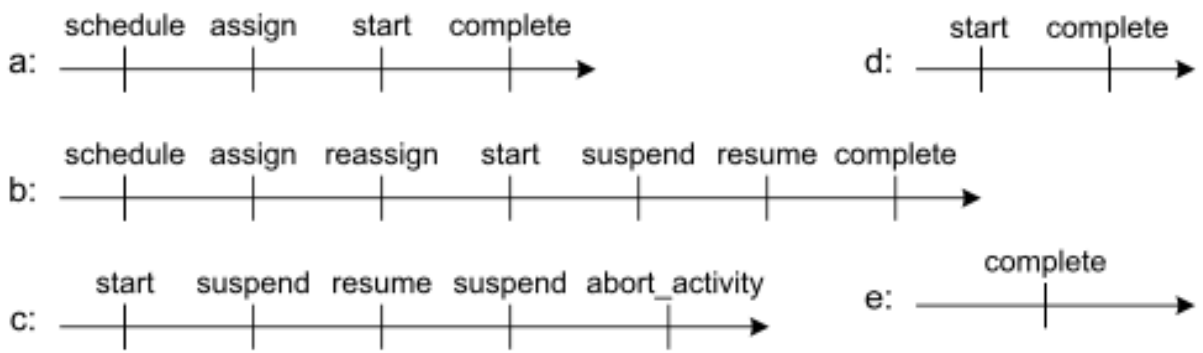
Timestamps should be **non-descending** in the event log
We assume a time domain $\mathcal{T}$, i.e., $\#time(e) \in \mathcal{T}$ for any $e \in \mathcal{E}$

Transaction type attribute

The transaction type attribute $\#trans(e)$ refers to the life-cycle of activities, we assume the following transactional life-cycle model



Example: transactional events for five different activity instances



Overlapping activity instances

We often refer to the event by its activity name - Technically this is not correct!!!!

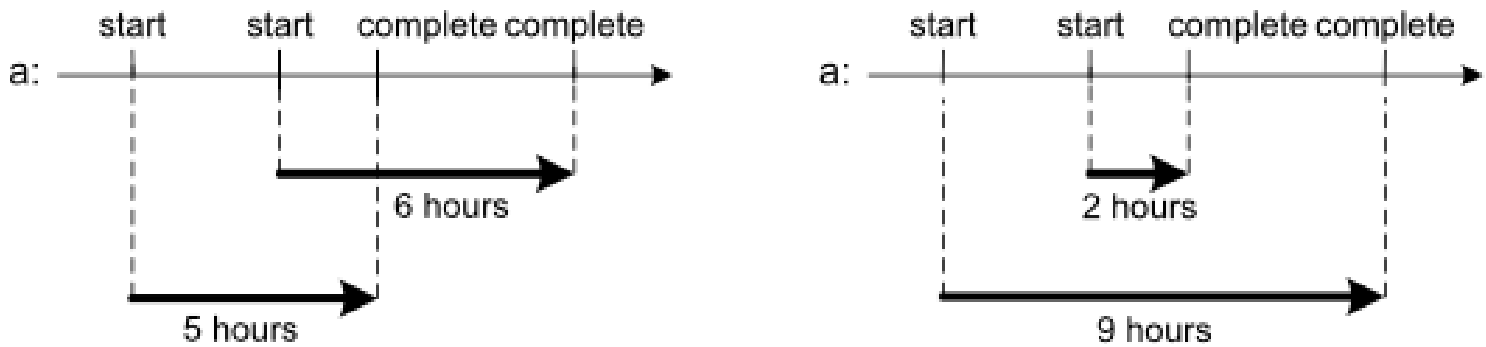
There may be many **events** that refer to the same **activity name**, within a case these events may refer:

- to the **same activity instance** (e.g., start and complete events)
- to **different activity instances** (e.g., in a loop)

This distinction is very important when measuring service times, waiting times, etc.

Example: Consider the scenario in which **the same activity is started twice** for the same case, i.e., two activity instances are running in parallel, and then one of them completes.

Did the activity that was started first complete or the second one?



They leave the same footprint in the event log!!!

Solving Correlation Problems

- by adding information to the log
  - When implementing systems, such information can easily be added to the logs; just provide an activity instance attribute to keep track of this
  - When dealing with existing systems this is not as simple as it seems, for example, when correlating messages between organizations there may be the need to scan the content of the message to find a suitable identifier (e.g., address or name)
- by using heuristics
  - One could just assume a first-in-first-out order and pick the first scenario!
  - One may introduce timeouts when the time between a start event and complete event is too long

Classifier

Some process mining techniques take into account the **transactional model** whereas others just consider **atomic events**

Sometimes we just want to focus:

- on complete events
- on withdrawals

This can be supported by filtering (e.g., removing events of a particular type) and by the concept of a **classifier**

A classifier is a function that maps the attributes of an event onto a label used in the resulting process model

Definition (Classifier)

For any event $e \in \mathcal{E}$ `event universe`, $\underline{e}$ is the name of the event

Classifier

In case $\underline{e} = \#activity(e)$ -> $\langle (a, b, c, d, e) \rangle$

In case $\underline{e} = \#activity(e), \#trans(e)$ -> $\langle (a, schedule)(a, assign)(a, start)(a, complete) \rangle$

We assume the classifier $\underline{e} = \#activity(e)$ as the default classifier
This is why we considered the activity attribute to be mandatory

Case, Trace and Event Log Definition

Definition (Case)

Let $\mathcal{C}$ be the case universe, i.e., the set of all possible case identifiers. Cases, like events, have attributes. For any case $c \in \mathcal{C}$ and name $n \in \mathcal{AN}$: $\#n(c)$ is the value of attribute n for case c ($\#n(c) = \perp$ if case c has no attribute named n). Each case has a special mandatory attribute trace, $\#trace(c) \in \mathcal{E}^*$. $\hat{c} = \#trace(c)$ is a shorthand for referring to the trace of a case.

Definition (Trace)

A trace is a finite sequence of events $\sigma \in \mathcal{E}^*$ such that each event appears only once, i.e., for $1 \leqslant i < j \leqslant |\sigma|$: $\sigma(i) \neq \sigma(j)$.

Definition (Event log)

An event log is a set of cases $\mathcal{L} \subseteq \mathcal{C}$ such that each event appears at most once in the entire log, i.e., for any $c_1, c_2 \in L$ such that $c_1 \neq c_2$: $\theta(\hat{c_1}) \cap \theta(\hat{c_2}) = \varnothing$.

If an event log contains timestamps, then the ordering in a trace should respect these timestamps, i.e., for any $c \in \mathcal{L}$, i and j such that $1 \leqslant i < j \leqslant |\hat{c}|$: $\#time(c(\hat{i})) \leqslant \#time(c(\hat{j}))$.

Events and cases are represented using unique identifiers

- An identifier $e \in \mathcal{E}$ refers to an event –> This is important as there may be many events having identical attributes, e.g., start events of some activity a may have been recorded for different cases and even within a case there may be multiple of such events
- An identifier $c \in \mathcal{C}$ refers to a case –> This is important as there may be different cases that followed the same path in the process

These identifiers are just a **technicality** that helps us to point to particular events and cases, **they do not need to exist in the original data source** and may be generated when extracting the data from different data sources

Formal Representation (s)

We can use the given formal representation to query the event log and use it as a starting point for analysis and reasoning

- $\{\#activity(e)|c \in \mathcal{L} \wedge e \in \hat{c}\}$ is the set of all activities appearing in log L
- $\{\#resource(e)|c \in \mathcal{L} \wedge e \in \hat{c} \wedge \#trans(e) = manualskip\}$ is the set of all resources that skipped an activity
- $\{a \in \mathcal{A}|c \in \mathcal{L} \wedge a = \#activity(\hat{c}(1)) \wedge a = \#activity(\hat{c}(|\hat{c}|))\}$ is the set of all activities that served as start and end activity for the same case

Depending on the attributes in the log, different types of analysis are possible

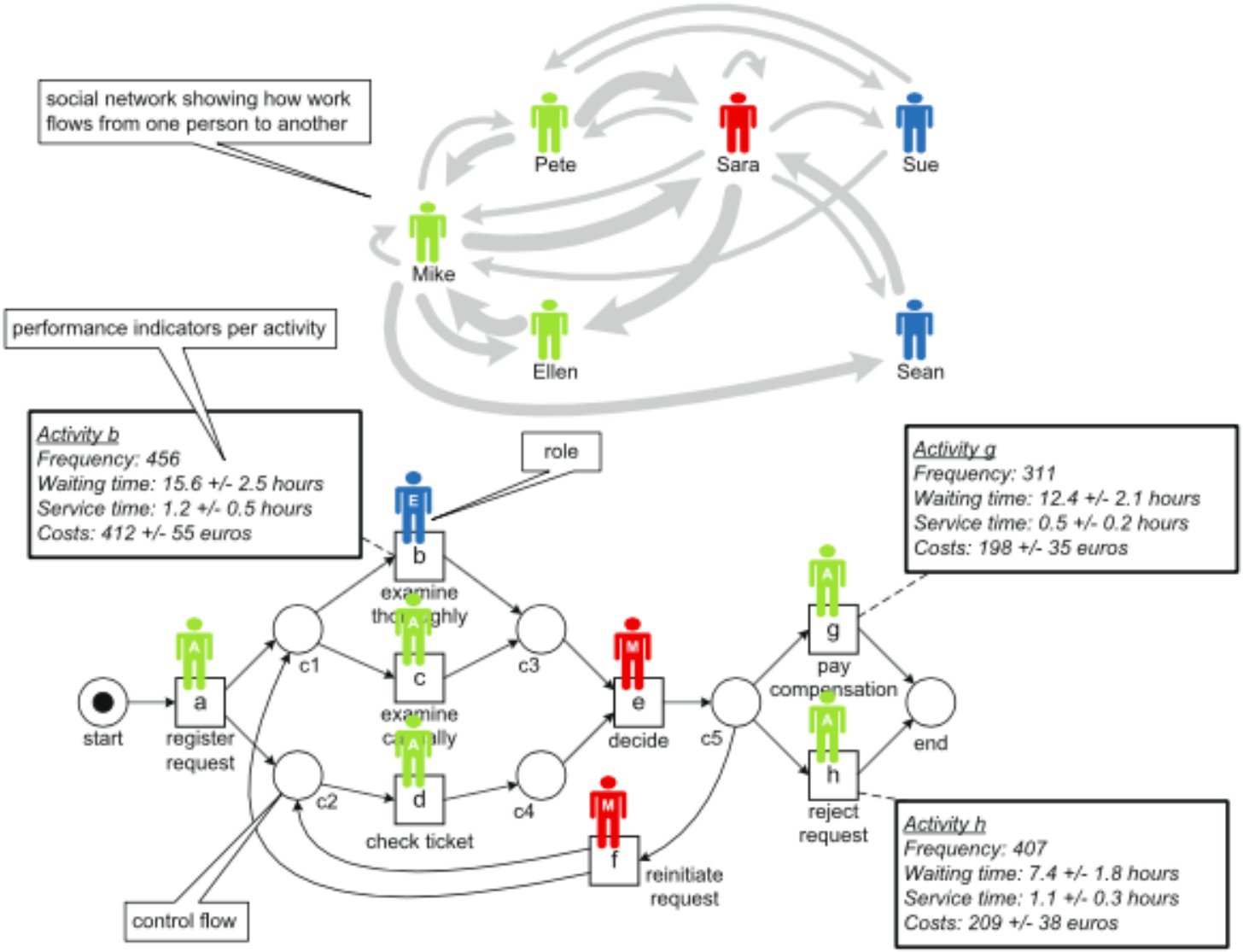
The **Petri net** can be discovered by just using the **activity attribute** (#activity(e))

To **measure durations of activities**, one needs to have a **transactional attribute** (#trans(e)) to distinguish start from completion, and **timestamps** (#time(e))

To **measure costs**, the **costs attribute** is used (#costs(e))

A **social network** can be discovered using the **resource attribute** (#resource(e))

Process Mining Types



Simple Event Log

Definition (Trace)

Let $\mathcal{A}$ be a set of activity names
A simple trace σ is a sequence of activities, i.e., $\sigma \in \mathcal{A}^*$
A simple event log $\mathcal{L}$ is a multi-set of traces over $\mathcal{A}$, i.e., $\mathcal{L} \in \mathbb{B}(\mathcal{A}^*)$

For example $(\langle a, b, c, d\rangle^3, \langle a, c, b, d\rangle^2, \langle a, e, d\rangle)$ defines a log containing 6 cases. In total there are 3 X 4 + 2 X 4 + 1 X 3 = 23 events

In a simple log there are no attributes, e.g., timestamps and resource information are abstracted from

Simple Event Log

Definition (Transforming an event log into a simple event log)

Let $\mathcal{L} \subseteq \mathcal{C}$ be an event log. Assume that a classifier has been defined: $\underline{e}$ is the name of event $e \in \mathcal{E}$. This classifier can also be applied to sequences, i.e.,
$\langle e_1, e_2, ..., e_n \rangle = \langle \underline{e_1}, \underline{e_2}, ..., \underline{e_n} \rangle$
$\underline{\mathcal{L}} = [\underline{(\hat{c})} | c \in \mathcal{L}]$ s the simple event log corresponding to $\mathcal{L}$

All cases in $\mathcal{L}$ are converted into sequences of (activity) names using the classifier

A case $c \in \mathcal{L}$ is an identifier from the case universe $\mathcal{C}$

$\hat{c} = \#trace(c) = \langle e_1, e_2, ..., e_n \rangle \in \mathcal{E}^*$ is the sequence of events executed for c

$\underline{\hat{c}} = \langle \underline{e_1}, \underline{e_2}, ..., \underline{e_n} \rangle$ maps these events onto (activity) names using the classifier

Simple Event Log

If we apply this transformation to the event log shown our previous example while assuming the default classifier ($\underline{e} = \#activity(e)$), then we obtain the event log

$$\underline{\mathcal{L}} = [\langle registerrequest, examinethoroughly, checkticket, decide, rejectrequest\rangle,$$
$$\langle registerrequest, checkticket, examinecasually, decide, paycompensation\rangle,$$
$$\langle registerrequest, examinecasually, checkticket, decide, reinitiaterequest,$$
$$examinethoroughly, checkticket, decide, paycompensation\rangle,$$
$$\langle registerrequest, checkticket, examinethoroughly, decide, rejectrequest\rangle, \ldots]$$

Another classifier could have been used to create a simple log, when using the classifier $\underline{e} = \#resource(e)$, the following log is obtained:

$$\underline{\mathcal{L}} = [\langle Pete, Sue, Mike, Sara, Pete\rangle, \langle Mike, Mike, Pete, Sara, Ellen\rangle,$$
$$\langle Pete, Mike, Ellen, Sara, Sara, Sean, Pete, Sara, Ellen\rangle, \langle Pete, Mike, Sean, Sara, Ellen\rangle, \ldots]$$

XES - eXtensible Event Stream

XES - eXtensible Event Stream

- Until 2010 the de facto standard for storing and exchanging event logs was MXML (Mining eXtensible Markup Language)
- In September 2010, the XES format was adopted by the **IEEE Task Force on Process Mining** and became the **de facto exchange format for process mining**
- On November 11th, 2016, the XES Standard has been officially published by the IEEE as an **official IEEE standard**
- See **www.xes-standard.org** for detailed information about the standard

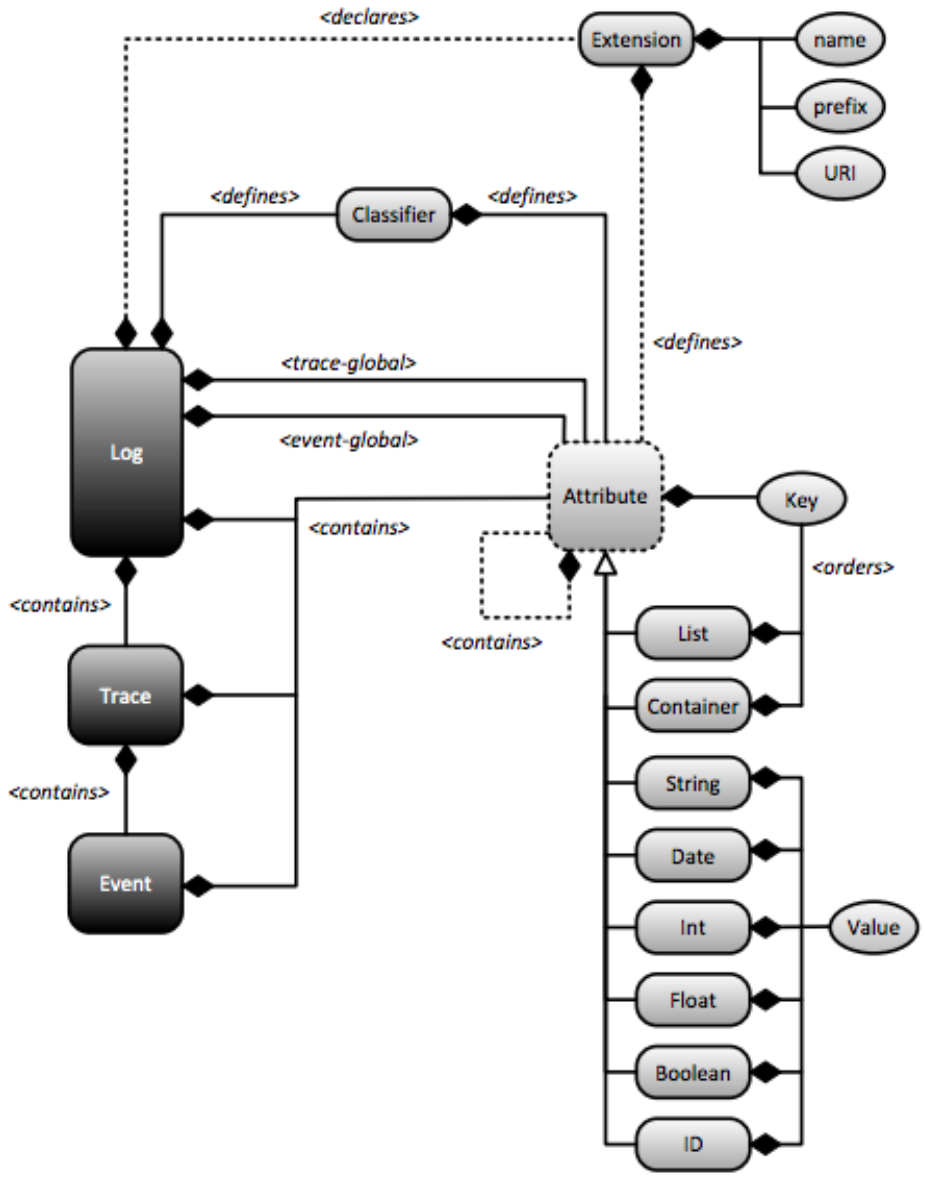
XES - eXtensible Event Stream

- The XES standard defines a grammar for a tag-based language whose aim is to provide designers of information systems with a unified and extensible methodology for capturing systems behaviors by means of event logs and event streams is defined in the XES standard
- An XML Schema describing the structure of an XES event log/stream and a XML Schema describing the structure of an extension of such a log/stream are included in this standard
- A basic collection of so-called XES extension prototypes that provide semantics to certain attributes as recorded in the event log/stream is included in this standard

XES - Principles

- **Simplicity** Use the simplest possible way to represent information. XES logs should be easy to parse and to generate, and they should be equally well human-readable. In designing this standard, care has been taken to take a pragmatic route wherever that benefits an ease of implementation.
- **Flexibility** The XES standard should be able to capture event logs from any background, no matter what the application domain or IT support of the observed process. Thus, XES aims to look beyond process mining and business processes, and strives to be a general standard for event log data.
- **Extensibility** It must be easy to add to the standard in the future. Extension of the standard should be as transparent as possible, while maintaining backward and forward compatibility. In the same vein, it must be possible to extend the standard for special requirements, e.g. for specific application domains, or for specific tool implementations.
- **Expressivity** While striving for a generic format, event logs serialized in XES should encounter as little loss of information as possible. Thus, all information elements must be strongly typed, and there must be a generic method to attach human-interpretable semantics to them.

XES meta model expressed in terms of a UML class diagram



XES meta model expressed in terms of a UML class diagram

- Log object, which contains all event information that is related to one specific process
- A log contains an arbitrary number (may be empty) of trace objects. Each trace describes the execution of one specific instance, or case, of the logged process
- Every trace contains an arbitrary number (may be empty) of event objects. Events represent atomic granules of activity that have been observed during the execution of a process.
- The log, trace, and event objects contain no information themselves. They only define the structure of the document. All information in an event log is stored in attributes. Logs, traces, and events each contain an arbitrary number of attributes. There are six types of elementary attributes, each defined by the type of data value they represent. Next to these elementary attributes, there are two types of collection attributes. For providing maximum flexibility in data storage, XES allows nested attributes, i.e. attributes can themselves have child attributes (note that this feature is required when using lists and/or containers).

XES classifier

The XES format makes event classification configurable and flexible, by introducing the concept of event classifiers. An event classifier assigns to each event an identity, which makes it comparable to other events (via their assigned identity).

Classifiers are defined via a set of attributes, from which the class identity of an event is derived. In its simplest form, an event classifier is defined by one attribute, and the value of that attribute would yield the class identity of an event.

< classifier name="Activity classifier" keys="name status" / >

Challenges when extracting event logs

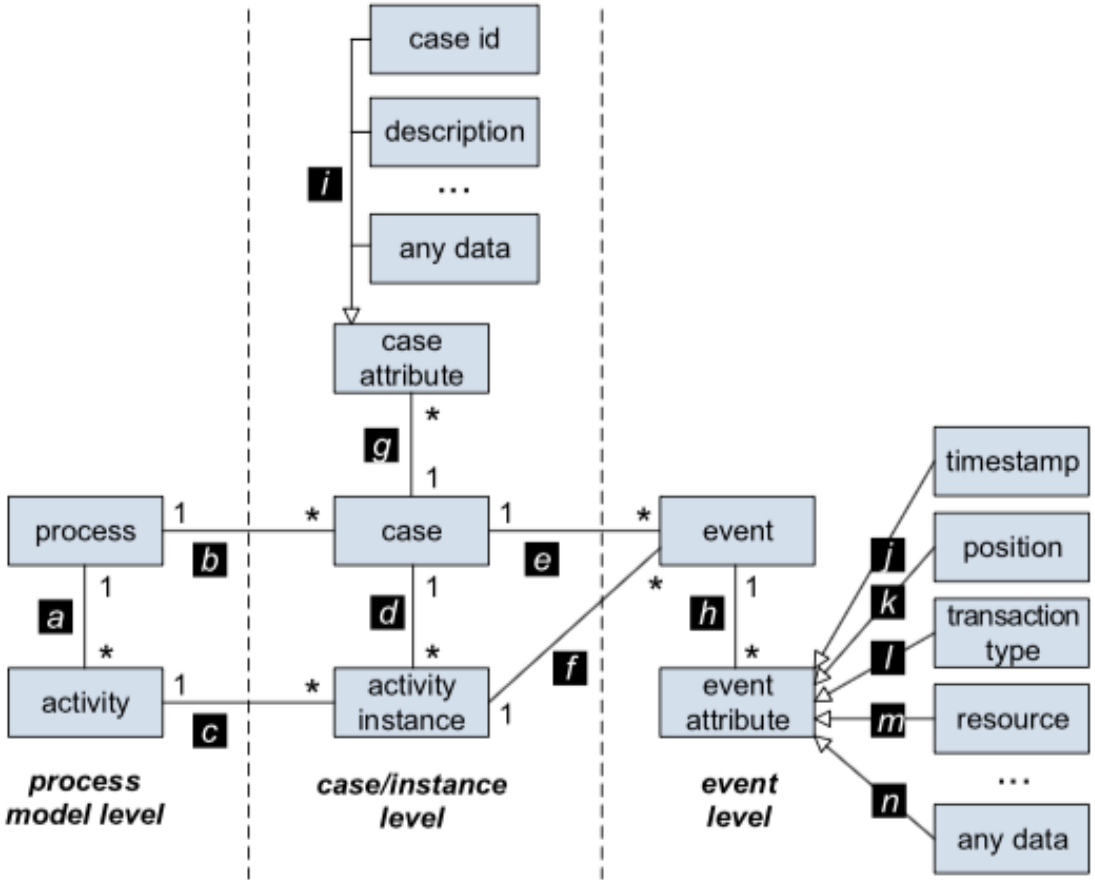
```
<?xml version="1.0" encoding="UTF-8" ?>
<log xes.version="2.0" xes.features="arbitrary-depth" xmlns="http://www.xes-standard.org
    /">
    <extension name="Concept" prefix="concept" uri="http://www.xes-standard.org/concept.
        xesext"/>
    <extension name="Time" prefix="time" uri="http://www.xes-standard.org/time.xesext"/>
    <global scope="trace">
        <string key="concept:name" value=""/>
    </global>
    <global scope="event">
        <string key="concept:name" value=""/>
        <date key="time:timestamp" value="1970-01-01T00:00:00.000+00:00"/>
        <string key="system" value=""/>
    </global>
    <classifier name="Activity" keys="concept:name"/>
    <classifier name="Another" keys="concept:name system"/>
    <float key="log attribute" value="2335.23"/>
    <trace>
        <string key="concept:name" value="Trace number one"/>
        <event>
            <string key="concept:name" value="Register client"/>
            <string key="system" value="alpha"/>
            <date key="time:timestamp" value="2009-11-25T14:12:45:000+02:00"/>
            <int key="attempt" value="23">
                <boolean key="tried hard" value="false"/>
            </int>
        </event>
        <event>
            <string key="concept:name" value="Mail rejection"/>
            <string key="system" value="beta"/>
            <date key="time:timestamp" value="2009-11-28T11:18:45:000+02:00"/>
        </event>
    </trace>
</log>
```

Challenges when extracting event logs

- **Correlation** - Events in an event log are grouped per case. This simple requirement can be quite challenging as it requires event correlation, i.e., events need to be related to each other.
- **Timestamps** - Events need to be ordered per case. Typical problems: only dates, different clocks, delayed logging.
- **Snapshots** - Cases may have a lifetime extending beyond the recorded period, e.g., a case was started before the beginning of the event log.
- **Scoping** - How to decide which tables to incorporate?
- **Granularity** - The events in the event log are at a different level of granularity than the activities relevant for end users.

Data Quality

Conceptualizing Event Logs



The given conceptualization discuss the key concepts without being distracted by the formalities and the technicalities of XES standard.

Data Quality Issues

- **Missing in log** - The entity exists (or existed) in reality, but was not recorded.
  - For example, an event (e.g., taking a blood sample) occurred but it was not captured by the information system.
- **Missing in reality** - The entity does not exist and never existed in reality, but was recorded.
  - For example, a scheduled doctor's appointment never took place due to an emergency, but it was recorded by the information system anyway.
- **Concealed in log** - The entity was recorded and exists (or existed) in reality, but it is hidden in a larger less structured data set.
  - For example, the same entity may appear multiple times in the event log. The scope of the data set may also be much larger than needed for analysis.

Type of problem (MIL, MIR, or CIL) versus the entity (CASE, AI, or EV) affected

Entity	Type of problem		
	Missing in log (*MIL*)	Missing in reality (*MIR*)	Concealed in log (*CIL*)
Case (*CASE*)	A case is missing in the event log, e.g., a customer order got flushed.	A case that never existed was added to the log, e.g., by inadvertently entering an improper identifier a fictive case is created.	A case is "hidden" in larger data set, e.g., customer orders, order lines, and deliveries with overlapping identifiers are intermingled in a single log.
Activity instance (*AI*)	An activity instance is missing in the event log, e.g., the start and complete of a production step are not related through an activity instance.	An activity instance that never existed was added to the log.	An activity instance is "hidden" in larger data set.
Event (*EV*)	An event is missing in the event log, e.g., a medical test was not recorded in the event log.	An event that never occurred was inadvertently added to the log, e.g., a check that was never conducted was recorded to feign compliance.	An event is "hidden" in larger data set, e.g., identifying a security breach from transactional data having a much broader scope.

TABLE 5.3

TABLE 5.4

12 Guidelines for Logging (I/II)

- **GL1** - Reference and attribute names should have clear semantics, i.e., they should have the same meaning for all people involved in creating and analyzing event data
- **GL2** - There should be a structured and managed collection of reference and attribute names
- **GL3** References should be stable (e.g., identifiers should not be reused or rely on the context)
- **GL4** Attribute values should be as precise as possible. If the value does not have the desired precision, then this should be indicated explicitly (e.g., through a qualifier)
- **GL5** Uncertainty with respect to the occurrence of the event or its references or attributes should be captured through appropriate qualifiers
- **GL6** Events should be at least partially ordered. The ordering of events may be stored explicitly (e.g., using a list) or implicitly through an attribute denoting the event?s timestamp

12 Guidelines for Logging (II/II)

- **GL7** If possible, also store transactional information about the event (start, complete, abort, schedule, assign, suspend, resume, withdraw, etc.)
- **GL8** Perform regularly automated consistency and correctness checks to ensure the syntactical correctness of the event log
- **GL9** Ensure comparability of event logs over time and different groups of cases or process variants
- **GL10** Do not aggregate events in the event log used as input for the analysis process
- **GL11** Do not remove events and ensure provenance. Reproducibility is key for process mining
- **GL12** Ensure privacy without losing meaningful correlations

Flattening Reality into Event Logs

Challenges when extracting event logs

In order to do process mining, events need to be related to cases!

This is natural as a process model describes the life-cycle of a case of a particular type, all activities in a conventional process model (independent of the notation used) correspond to status changes of such a case

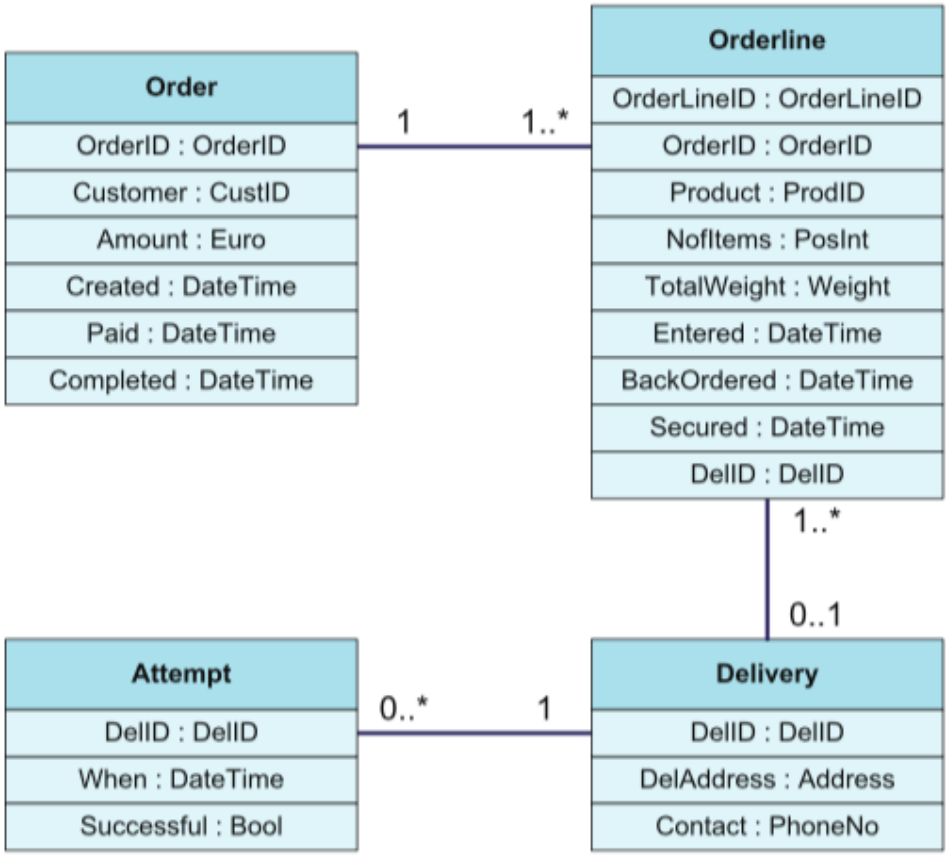
$$\Downarrow$$

We will refer to such process models as **flat models**

$$\Downarrow$$

However, it is important to realize that real-life processes are not flat.

Flattening Reality - Example



Class diagram showing the relations between orders, order lines, deliveries, and delivery attempts

Flattening Reality - Example

Order					
OrderID	Customer	Amount	Created	Paid	Completed
91245	John	100	28-11-2011:08.12	02-12-2011:13.45	05-12-2011:11.33
91561	Mike	530	28-11-2011:12.22	03-12-2011:14.34	05-12-2011:09.32
91812	Mary	234	29-11-2011:09.45	02-12-2011:09.44	04-12-2011:13.33
92233	Sue	110	29-11-2011:10.12	null	null
92345	Kirsten	195	29-11-2011:14.45	02-12-2011:13.45	null
92355	Pete	320	29-11-2011:16.32	null	null
...	...	...	...	...	...

Flattening Reality - Example II

Orderline

OrderLineID	OrderID	Product	NoItems	TotalWeight	Entered	BackOrdered	Secured	DellID
112345	91245	iPhone 4G	1	0.250	28-11-2011:08.13	null	28-11-2011:08.55	882345
112346	91245	iPod nano	2	0.300	28-11-2011:08.14	28-11-2011:08.55	30-11-2011:09.06	882346
112347	91245	iPod classic	1	0.200	28-11-2011:08.15	null	29-11-2011:10.06	882345
112448	91561	iPhone 4G	1	0.250	28-11-2011:12.23	null	28-11-2011:12.59	882345
112449	91561	iPod classic	1	0.200	28-11-2011:12.24	28-11-2011:16.22	null	null
112452	91812	iPhone 4G	5	1.250	29-11-2011:09.46	null	29-11-2011:10.58	882346
...	...	...	...	...	...	...	...	...

Flattening Reality - Example III

Delivery		
DelID	DelAddress	Contact
882345	5513VJ-22a	0497-2553660
882346	5513XG-45	040-2298761
...	...	...

Attempt		
DelID	When	Successful
882345	05-12-2011:08.55	false
882345	06-12-2011:09.12	false
882345	07-12-2011:08.56	true
882346	05-12-2011:08.43	true
...	...	...

Challenges when extracting event logs

Clearly the timestamps in the four tables correspond to events related to the ?overall ordering and delivery? process

However, when creating an event log, **each event needs to be associated to a particular case** –> Therefore, we need to flatten the four tables into one table with a ?case id? column

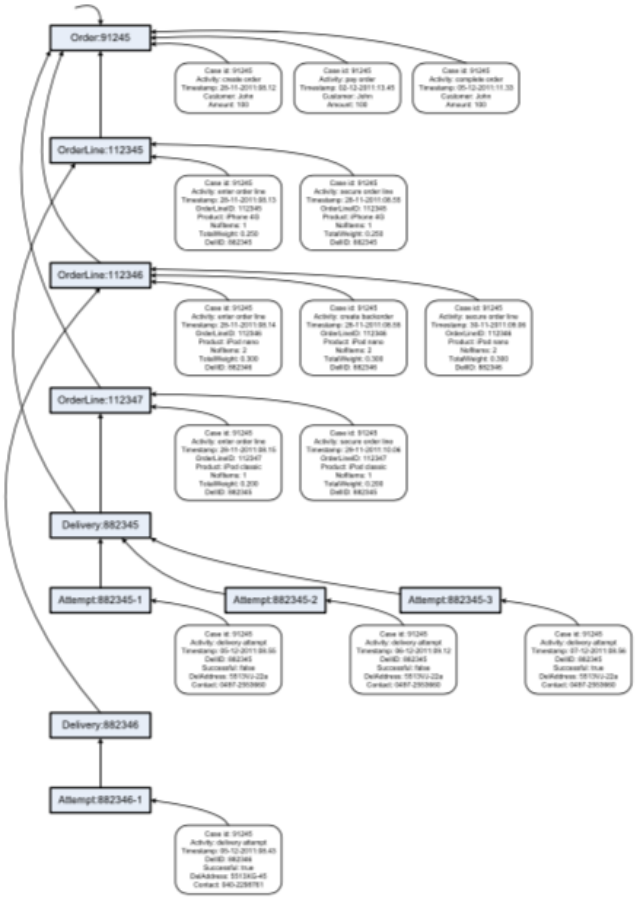
However, one can choose from four types of cases: orders, order lines, deliveries, and attempts. Any record in one of the four tables potentially corresponds to a case

$\Downarrow$

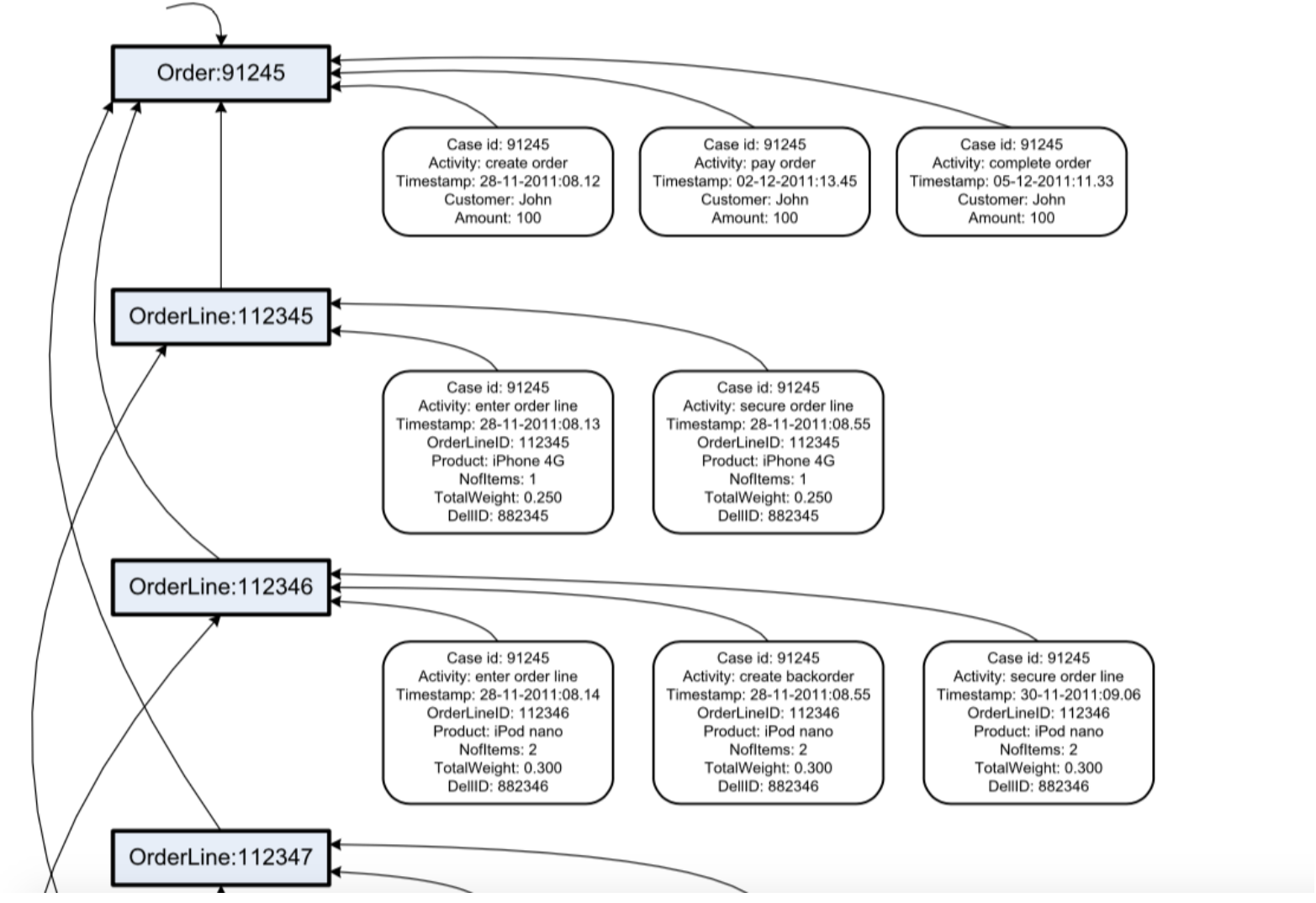
Which one to choose?

Flattening Reality - Example IV

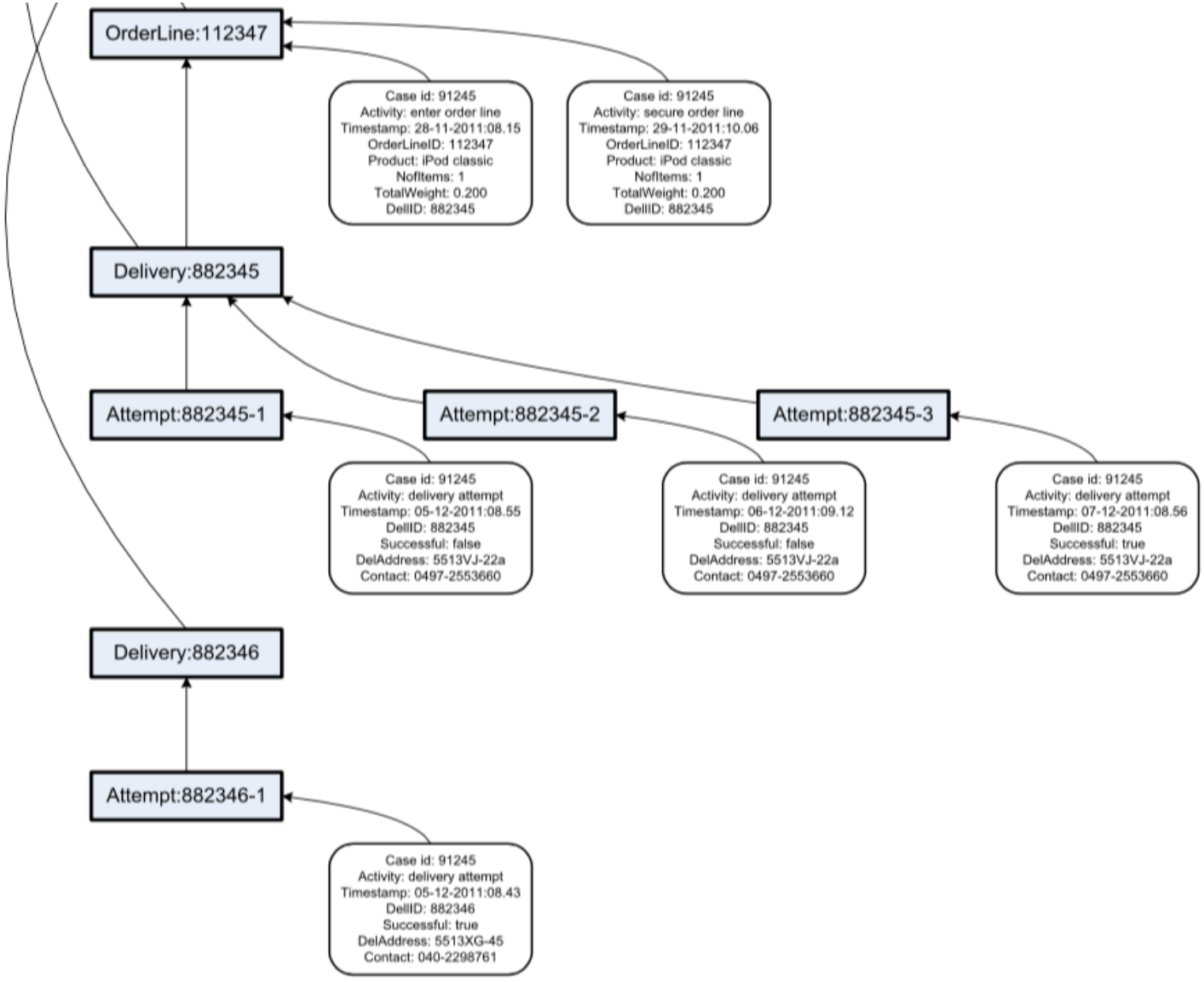
All events that can be related to order 91245. The 14 rounded rectangles correspond to events associated to case 91245. The squared rectangles represent records in one of the four tables



Flattening Reality - Example IV (part 1)



Flattening Reality - Example IV (part 2)



Flattening Reality - Example V

Events extracted from all four tables using order records from the Order table as a starting point –> This is a possible way to flatts the original database consisting of four tables.

The flattened event log is like a view on the complete data set

Alternative views are possible

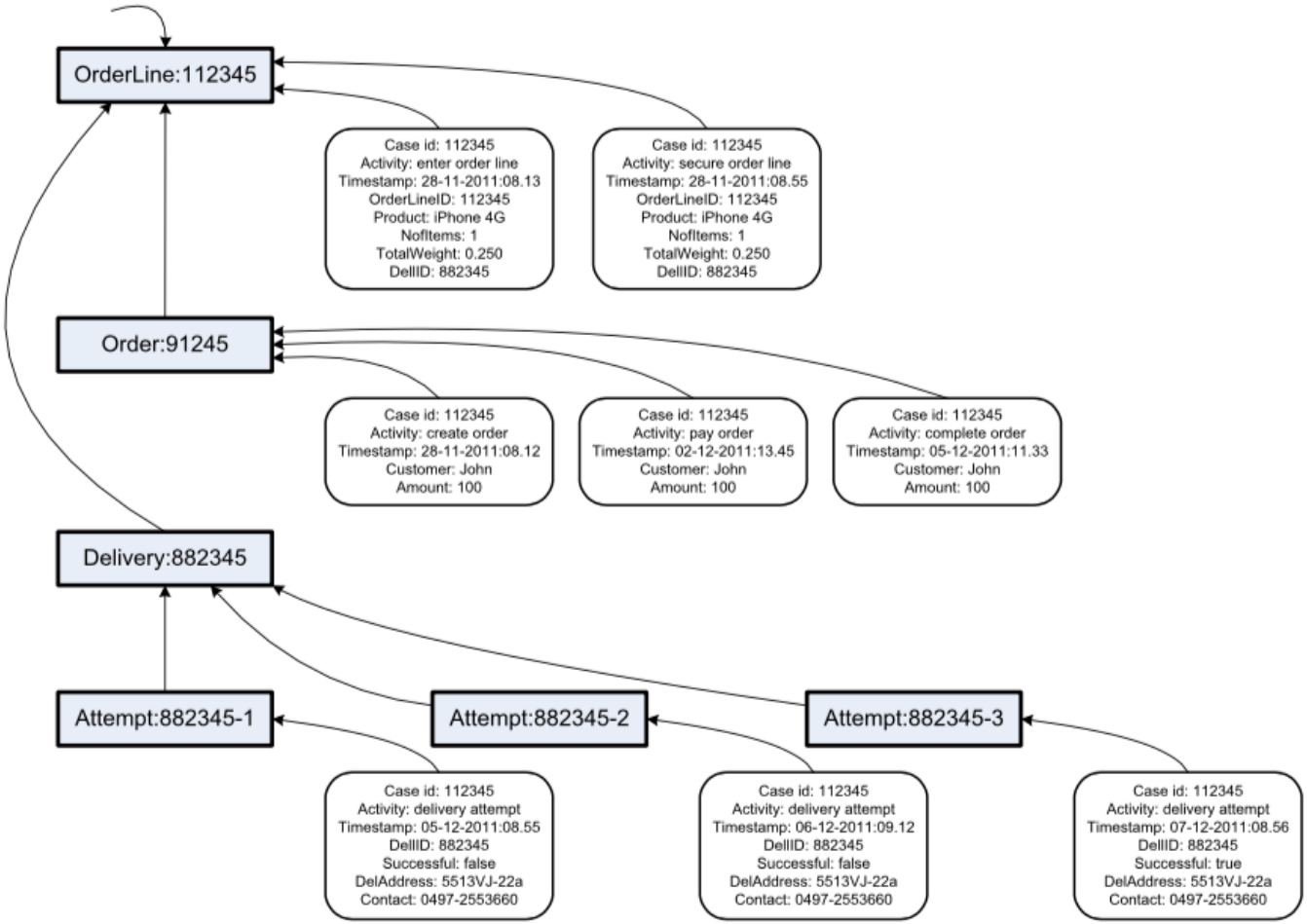
Attempt			
Case id	Activity	Timestamp	Other attributes
91245	create order	28-11-2011:08.12	Customer: John, Amount: 100
91245	enter order line	28-11-2011:08.13	OrderLineID: 112345, Product: iPhone 4G, NofItems: 1, TotalWeight: 0.250, DellID: 882345
91245	enter order line	28-11-2011:08.14	OrderLineID: 112346, Product: iPod nano, NofItems: 2, TotalWeight: 0.300, DellID: 882346
91245	enter order line	28-11-2011:08.15	OrderLineID: 112347, Product: iPod classic, NofItems: 1, TotalWeight: 0.200, DellID: 882345
91245	secure order line	28-11-2011:08.55	OrderLineID: 112345, Product: iPhone 4G, NofItems: 1, TotalWeight: 0.250, DellID: 882345
91245	create backorder	28-11-2011:08.55	OrderLineID: 112346, Product: iPod nano, NofItems: 2, TotalWeight: 0.300, DellID: 882346
91245	secure order line	29-11-2011:10.06	OrderLineID: 112347, Product: iPod classic, NofItems: 1, TotalWeight: 0.200, DellID: 882345
91245	secure order line	30-11-2011:09.06	OrderLineID: 112346, Product: iPod nano, NofItems: 2, TotalWeight: 0.300, DellID: 882346
91245	pay order	02-12-2011:13.45	Customer: John, Amount: 100
91245	delivery attempt	05-12-2011:08.43	DellID: 882346, Successful: true, DelAddress: 5513XG-45, Contact: 040-2298761
91245	delivery attempt	05-12-2011:08.55	DellID: 882345, Successful: false, DelAddress: 5513VJ-22a, Contact: 0497-2553660
91245	complete order	05-12-2011:11.33	Customer: John, Amount: 100
91245	delivery attempt	06-12-2011:09.12	DellID: 882345, Successful: false, DelAddress: 5513VJ-22a, Contact: 0497-2553660
91245	delivery attempt	07-12-2011:08.56	DellID: 882345, Successful: true, DelAddress: 5513VJ-22a, Contact: 0497-2553660
91561	create order	28-11-2011:12.22	Customer: Mike, Amount: 530
91561	enter order line	28-11-2011:12.23	OrderLineID: 112448, Product: iPhone 4G, NofItems: 1, TotalWeight: 0.250, DellID: 882345
...	...	...	...
...	...	...	...

Flattening Reality - Example V (highlight)

Attempt			
Case id	Activity	Timestamp	Other attributes
91245	create order	28-11-2011:08.12	Customer: John, Amount: 100
91245	enter order line	28-11-2011:08.13	OrderLineID: 112345, Product: iPhone 4G, NofItems: 1, TotalWeight: 0.250, DellID: 882345
91245	enter order line	28-11-2011:08.14	OrderLineID: 112346, Product: iPod nano, NofItems: 2, TotalWeight: 0.300, DellID: 882346
91245	enter order line	28-11-2011:08.15	OrderLineID: 112347, Product: iPod classic, NofItems: 1, TotalWeight: 0.200, DellID: 882345
91245	secure order line	28-11-2011:08.55	OrderLineID: 112345, Product: iPhone 4G, NofItems: 1, TotalWeight: 0.250, DellID: 882345
91245	create backorder	28-11-2011:08.55	OrderLineID: 112346, Product: iPod nano, NofItems: 2, TotalWeight: 0.300, DellID: 882346
91245	secure order line	29-11-2011:10.06	OrderLineID: 112347, Product: iPod classic, NofItems: 1, TotalWeight: 0.200, DellID: 882345
91245	secure order line	30-11-2011:09.06	OrderLineID: 112346, Product: iPod nano, NofItems: 2, TotalWeight: 0.300, DellID: 882346

Various selections of events can be used!!!

All events that can be related to order line 112345



2D vs 3D

Although it is important to view business processes in 3-D, we often need to resort to 2-D models for a variety of reasons.

- the data sources provided may only allow for a 2-D view, e.g., only one table is provided as input
- users expect process models in terms of classical 2-D process modeling languages such as BPMN, UML ADs, Statecharts, BPEL, YAWL, WF-nets, and EPCs
- most process mining techniques require flattening the data