**Physical Based models**

**Getting the Data**

Process mining is impossible without proper event logs. Depending on the process mining technique used, these requirements may vary. 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. When merging and extracting data, both syntax and semantics play an important role.

Process mining, like any other data-driven analysis approach, needs to deal with data quality problems.

**5.1 Data Sources**

the concept of process mining. The idea is to analyze event data from a process-oriented perspective. The goal of process mining is to answer questions about operational processes.

Examples are:

• What *really* happened in the past?

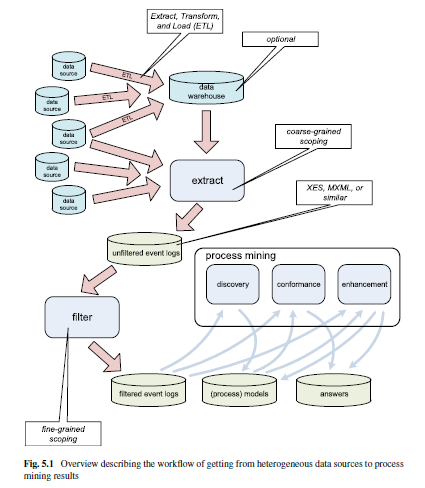
• Why did it happen?

• 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?



The reality is that event data is typically scattered over different data sources and often quite some efforts are needed to collect the relevant data.

Data may be scattered due to technical or organizational reasons.

For example, there may be legacy systems holding crucial data or information systems used only at the departmental level. For cross-organizational process mining, e.g., to analyze supply chains, data may even be scattered over multiple organizations.

Data sources may be structured and well-described by meta data. Unfortunately, in many situations the data is

unstructured or important meta data is missing. Data may originate from web pages, emails, PDF documents, scanned text, screen scraping, etc. There is no point in trying to exhaustively extract event logs

from thousands of tables and other data sources. Data extraction should be driven by questions rather than the availability of lots of data.

In the context of BI and data mining, the phrase *“Extract, Transform, and Load”* (ETL) is used to describe the process that involves:

(a) *extracting* data from outside sources,

(b) *transforming* it to fit operational needs (dealing with syntactical and semantical issues while ensuring predefined quality levels),

(c) *loading* it into the target system,

e.g., a data warehouse or relational database.

A *data warehouse* is a single logical repository of an organization’s transactional and operational data.

The data warehouse does not produce data but simply taps off data from operational systems. The goal is to unify information such that it can be used for reporting, analysis, forecasting, etc.

If a data warehouse already exists, it most likely holds valuable input for process mining. However, many organizations do not have a good data warehouse. The warehouse may contain only a subset of the information needed for end-to-end process mining, e.g., only data related to customers is stored. Moreover, if a data warehouse is present, it does not need to be process oriented.

For example, the typical warehouse data used for *Online Analytical Processing* (OLAP) does not provide much process-related information.

OLAP tools are excellent for viewing multidimensional data from different angles, drilling down, and for creating all kinds of reports. However, OLAP tools do not require the storage of business events and their ordering.

The data sets used by the mainstream data mining approaches also do not store such information.

For example, a decision tree learner can be applied to any table consisting of rows (instances) and columns (variables).

Typical formats to store event logs are *XES* (eXtensible Event Stream) and *MXML* (Mining eXtensible Markup Language).

The problem of converting “3-D data” into “2-D event logs”, i.e., events are projected onto the desired process model.

Consider for example the data in a hospital. One may be interested in the discovery of patient flows, i.e., typical diagnosis and treatment paths.

However, one may also be interested in optimizing the workflow within the radiology department.

Both questions require different event logs, although some events may be shared among the two required event logs. Once an event log is created, it is typically *filtered*.

Filtering is an iterative process. *Coarse-grained scoping* was done when extracting the data into an event log.

**5.2 Event Logs**

The table shows events related to the handling of requests for compensation.

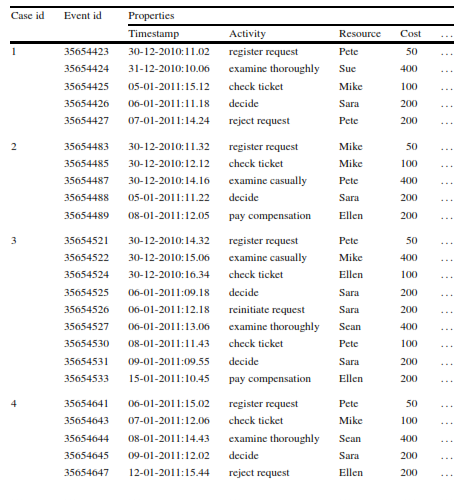
We assume that an event log contains data related to a *single process*, i.e., the first coarse-grained scoping step in Fig. 5.1 should make sure that all events can be related to this process. Moreover, each event in the log needs to refer to a *single process instance*, often referred to as *case*. In Table 5.1, each request corresponds to a case, e.g., case 1. We also assume that events can be related to some *activity*.

In Table 5.1, events refer to activities like *register request*, *check ticket*, and *reject*.

These assumptions are quite natural in the context of process mining. All mainstream process modeling notations, specify a process as a collection of activities such that the life-cycle of a single instance is described. Hence, the “case id” and “activity” columns in Table 5.1 represent the bare minimum for process mining. Moreover, events within a case need to be ordered.

For example, event 35654423 (the execution of activity *register request* for Case 1) occurs before event 35654424 (the execution of activity *examine thoroughly* for the same case). Without ordering information it is of course impossible to discover causal dependencies in process models.

Table 5.1 also shows additional information per event. For example, all events have a *timestamp* (i.e., date and time information such as “30-12-2010:11.02”).

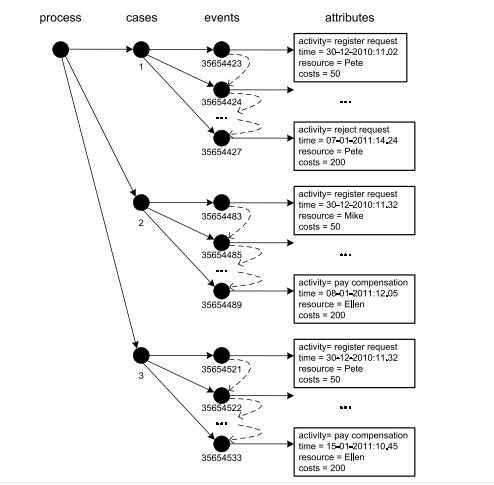


Using this ﬁgure we can list our assumptions about event logs.

• A *process* consists of *cases*.

• A case consists of *events* such that each event relates to precisely one case.

• Events within a case are *ordered*.

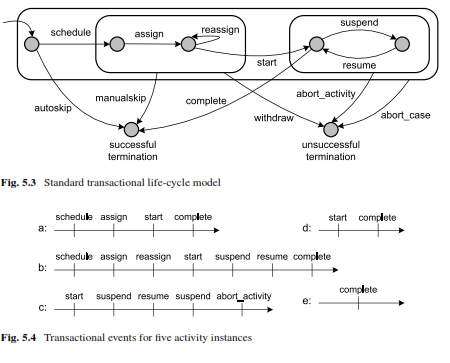


• Events can have *attributes*. Examples of typical attribute names are activity, time, costs, and resource.

Not all events need to have the same set of attributes. However, typically, events referring to the same activity have the same set of attributes.

To be able to reason about logs and to precisely specify the requirements for event logs, we formalize the various notions.

**Deﬁnition 5.1** (Event, attribute) Let 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 particular person, has associated costs, etc. Let *AN* be a set of attribute names. For any event



e ∈ E and name n ∈ *AN*,# (e) is the value of attribute n for event e. If event e does

not have an attribute named n, then # n n

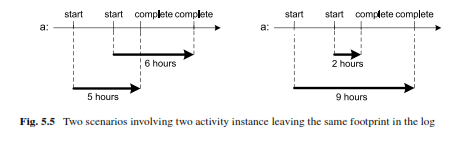
(e) =⊥(null value). For convenience we assume the following standard attributes:

• # (e) is the *activity* associated to event e.

• # *activity*

(e) is the *timestamp* of event e.

• # • # *Time resource Trans* (e) is the *resource* associated to event e. (e) is the *transaction type* associated to event e, examples are schedule, start, complete, and suspend.



**5.3 XES**

The de facto standard for storing and exchanging event logs was *MXML* (Mining eXtensible Markup Language).

MXML emerged in 2003 and was later adopted by the process mining tool ProM.

Using MXML it is possible to store event logs such as the one shown in Table 5.1 using an XML-based syntax. *ProMimport* is a tool supporting the conversion of different data sources to MXML,

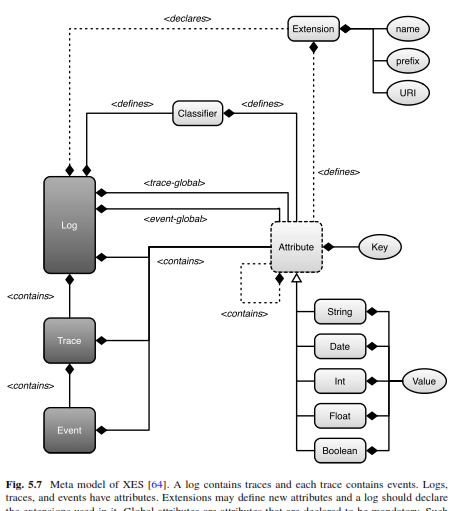
e.g., MS Access, Aris PPM, CSV, Apache, Adept, PeopleSoft, Subversion, SAP R/3, Protos, CPN Tools, Cognos, and Staffware. MXML has a standard notation for storing timestamps, resources, and transaction types.

*SA-MXML* (Semantically Annotated Mining eXtensible Markup Language) is a semantic annotated version of the MXML format used by the ProM framework. SA-MXML incorporates references between elements in logs and concepts in ontologies.

For example, a resource can have a reference to a concept in an ontology describing a hierarchy of roles, organizational entities, and positions. To realize these semantic annotations, existing XML elements were interpreted in a new manner.

Other extensions were realized in a similar manner.

XES is the successor of MXML. Based on many practical experiences with MXML, the XES format has been made less restrictive and truly extendible.



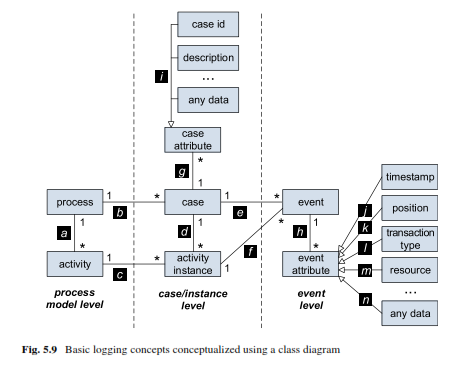
• The *concept extension* deﬁnes the *name* attribute for traces and events.

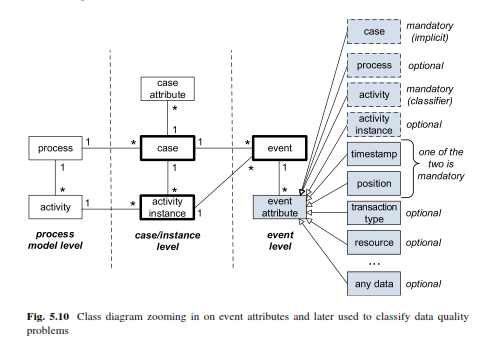
Note that the example XES ﬁle indeed uses *concept:name* attributes for traces and events.

**5.4 Data Quality**

From a practical point of view *data quality* is of the utmost importance for the success of process mining. If event data is missing or cannot be trusted, then the results of process mining are less valuable.

The conceptualization is used to systematic classify data quality problems.





**5.5 Flattening Reality into Event Logs**

In order to do process mining, events need to be related to cases. As indicated before, 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. We will refer to such process models as *ﬂat models*. In this book, we adopt this (often hidden) assumption associated to all mainstream process modeling notations. However, it is important to realize that *real-life processes are not ﬂat*. We use a simple example to illustrate this.

Consider the class diagram shown in Fig. 5.11 describing a database consisting of four tables. Table *Order* contains information about orders. For example, each record in the *Order* table has a unique order number, refers to a customer, and has an associated amount. Multiple products can be ordered in one order. Therefore,

