Chapter 9
Mining Additional Perspectives

Whereas the main focus of process discovery is on the control-flow perspective, event logs may contain a wealth of information relating to other perspectives such as the organizational perspective, the case perspective, and the time perspective. Therefore, we now shift our attention to these other perspectives. Organizational mining can be used to get insight into typical work patterns, organizational structures, and social networks. Timestamps and frequencies of activities can be used to identify bottlenecks and diagnose other performance related problems. Case data can be used to better understand decision-making and analyze differences among cases. Moreover, the different perspectives can be merged into a single model providing an integrated view on the process. Such an integrated model can be used for "what if" analysis using simulation.

9.1 Perspectives

Thus far the focus of this book was on control-flow, i.e., the ordering of activities. The chapters on process discovery and conformance checking often used a so-called "simple event log" as a starting point (see Definition 5.4). However, as discussed in Chap. 5, event logs typically contain much more information. Events and cases can have any number of attributes (see Definitions 5.1 and 5.3). The extension mechanism of XES illustrates how such attributes can be structured and stored. Moreover, as stressed in Sect. 2.2, process mining is not limited to the control-flow perspective. Therefore, we now focus on adding some of the other perspectives.

Figure 9.1 shows a typical scenario. The starting point is an event log and some initial process model. Note that the process model may have been constructed manually or discovered through process mining. Important is that the process model and event log are connected. In Sect. 8.5.3, we showed that the replay approaches used in the context of conformance checking can be used to tightly couple model and log. As discussed using Fig. 8.14, activity instances discovered during replay connect modeled activities to recorded events. This way attributes of events (resources,

W. van der Aalst, *Process Mining*, DOI 10.1007/978-3-662-49851-4_9

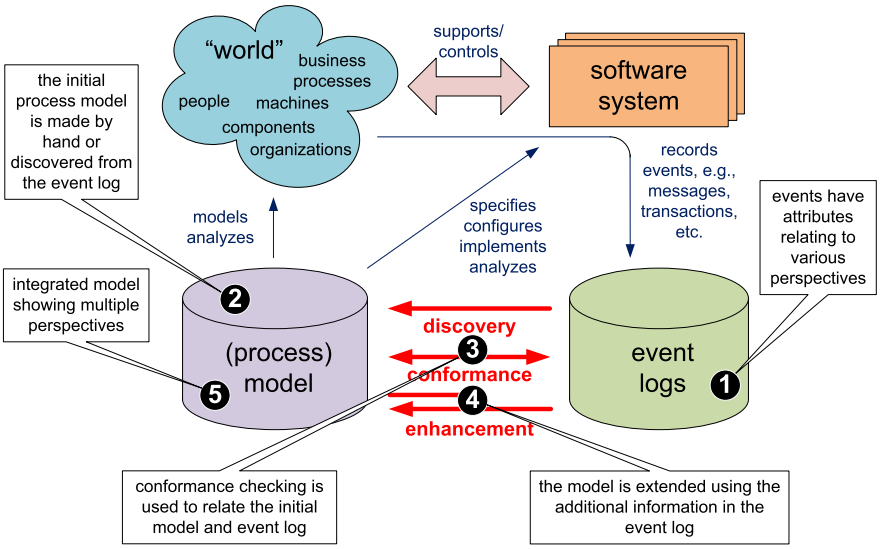

Fig. 9.1 The organizational, case, and time perspectives can be added to the original control-flow model using attributes from the event log

timestamps, costs, etc.) can be used to extend the initial model. For example, information about service or waiting times extracted from the event log can be added to the model. After adding the different perspectives, an *integrated* process model is obtained.

Figure 9.1 lists the three main types of process mining: *discovery*, *conformance*, and *enhancement*. Let us focus on the third type of process mining. Enhancement aims to extend or improve an existing process model using information about the actual process recorded in some event log. One type of enhancement is *repair* as discussed in Sect. 8.5.1. Here, we devote our attention to other type of enhancement: *extension*. Through extension we add a new perspective to the process model by cross-correlating it with the log.

In the remainder, we show some examples of log-based model extension. Section 9.3 discusses various process mining techniques related to the organizational perspective. Here, information about resources is used to analyze working patterns and to see how work "flows" through an organization. Extensions based on the time perspective are discussed in Sect. 9.4. When events bear timestamps it is possible to discover bottlenecks, measure service levels, monitor the utilization of resources, and predict the remaining processing time of running cases. Section 9.5 focuses on other attributes and their effects on decision making. This section illustrates that classical data mining techniques such as decision tree learning can be used to extend a process model with the case perspective. The different perspectives can be merged into a single integrated process model. Section 9.6 shows how such an integrated model can be constructed and used. For instance, a complete simulation model can be mined and subsequently used for "what if" analysis.

Table 9.1 A fragment of some event log: each line corresponds to an event

Case id	Event id	Properties				
		Time	Activity	Trans	Resource	Cost
1	35654423	30-12-2010:11.02	register request	start	Pete	
	35654424	30-12-2010:11.08	register request	complete	Pete	50
	35654425	31-12-2010:10.06	examine thoroughly	start	Sue	
	35654427	31-12-2010:10.08	check ticket	start	Mike	
	35654428	31-12-2010:10.12	examine thoroughly	complete	Sue	400
	35654429	31-12-2010:10.20	check ticket	complete	Mike	100
	35654430	06-01-2011:11.18	decide	start	Sara	
	35654431	06-01-2011:11.22	decide	complete	Sara	200
	35654432	07-01-2011:14.24	reject request	start	Pete	
	35654433	07-01-2011:14.32	reject request	complete	Pete	200
2	35654483	30-12-2010:11.32	register request	start	Mike	
	35654484	30-12-2010:11.40	register request	complete	Mike	50
	35654485	30-12-2010:12.12	check ticket	start	Mike	
	35654486	30-12-2010:12.24	check ticket	complete	Mike	100
	35654487	30-12-2010:14.16	examine casually	start	Pete	
	35654488	30-12-2010:14.22	examine casually	complete	Pete	400
	35654489	05-01-2011:11.22	decide	start	Sara	
	35654490	05-01-2011:11.29	decide	complete	Sara	200
	35654491	08-01-2011:12.05	pay compensation	start	Ellen	
	35654492	08-01-2011:12.15	pay compensation	complete	Ellen	200
...	...	...	...	...	...	...

9.2 Attributes: A Helicopter View

Before discussing approaches to discover the resource, time, and case perspectives, we provide another example showing the kind of information one can find in a typical event log. Table 9.1 shows a small fragment of a larger event log. Compared to earlier examples, each event now also has a *transaction type*. Consider, for example, the first two events in Table 9.1. The first event refers to the *start* of an activity instance, whereas the second event refers to the *completion* of this instance. By taking the difference between the timestamps of both events, it can be derived that Pete worked for six minutes on case 1 when registering the request of the customer. Only events with transaction type *complete* have a cost attribute. Note that Sue and Mike are both working on the same case at the same time, because activities *examine thoroughly* and *check ticket* for case 1 are overlapping.

Table 9.2 shows the *case attributes* stored in the event log. These are attributes that refer to the case as a whole rather than an individual event (see Definition 5.3). Case 1 is a request initiated by customer *Smith*. This customer has an identification

Table 9.2 Attributes of cases

Case id	Custid	Name	Type	Region	Amount
1	9911	Smith	gold	south	989.50
2	9915	Jones	silver	west	546.00
3	9912	Anderson	silver	north	763.20
4	9904	Thompson	silver	west	911.70
5	9911	Smith	gold	south	812.10
6	9944	Baker	silver	east	788.00
7	9944	Baker	silver	east	792.80
8	9911	Smith	gold	south	544.70
...	...	...	...	...	...

number *9911*. Customer Smith is a *gold* customer in region *south*. The amount of compensation requested is € 989.50. Cases 5 and 8 are also initiated by the same customer. Case 2 is initiated by silver customer *Jones* from region *west*. This customer claimed an amount of € 546.00.

Each of the events implicitly refers to attributes of the corresponding case. For instance, event 35654483 implicitly refers to silver customer Jones because the event is executed for case 2. In Chap. 5, we formalized the notion of an event log and event attributes. Consider for example $e = 35654431$ and some of its attributes: $\#_{case}(e) = 1$, $\#_{activity}(e) = decide$, $\#_{time}(e) = 06\text{-}01\text{-}2011{:}11.22$, $\#_{resource}(e) = Sara$, $\#_{trans}(e) = complete$, $\#_{cost}(e) = 200$, $\#_{custid}(e) = 9911$, $\#_{name}(e) = Smith$, $\#_{type}(e) = gold$, $\#_{region}(e) = south$, and $\#_{amount}(e) = 989.50$. For process discovery, we ignored most of these attributes. This chapter will show how to use these attributes to create an integrated model covering different perspectives.

A first step in any process mining project is to get a feeling for the process and the data in the event log. The so-called *dotted chart* provides a helicopter view of the process [129]. In a dotted chart, each event is depicted as a dot in a two dimensional plane as shown in Fig. 9.2. The horizontal axis represents the *time* of the event. The vertical axis represents the *class* of the event. To determine the class of an event we use a *classifier* as described in Definition 5.2. A classifier is a function that maps the attributes of an event onto a label, $\underline{e}$ is the *class* of the event. An example of a classifier is $\underline{e} = \#_{case}(e)$, i.e., the case id of the event. Other examples are $\underline{e} = \#_{activity}(e)$ (the name of the activity being executed) and $\underline{e} = \#_{resource}(e)$ (the resource triggering the event). In this particular example, $\underline{e} = \#_{region}(e)$ would be a classifier mapping the event onto the region of the customer.

Every line in the dotted chart shown in Fig. 9.2 refers to a class, e.g., if the classifier $\underline{e} = \#_{resource}(e)$ is used, then every line corresponds to a resource. The dots on such a line describe the events belonging to this class, e.g., all events executed by a particular resource. The time dimension can be *absolute* or *relative*. If time is relative, the first event of each case takes place at time zero. Hence, the horizontal position of the dot depends on the time passed since the first event for the same case. The time dimension can be *real* or *logical*. For real time, the actual timestamp is used. For logical time, events are simply enumerated without considering the actual timestamps: only their ordering is taken into account. The first event has time 0, the

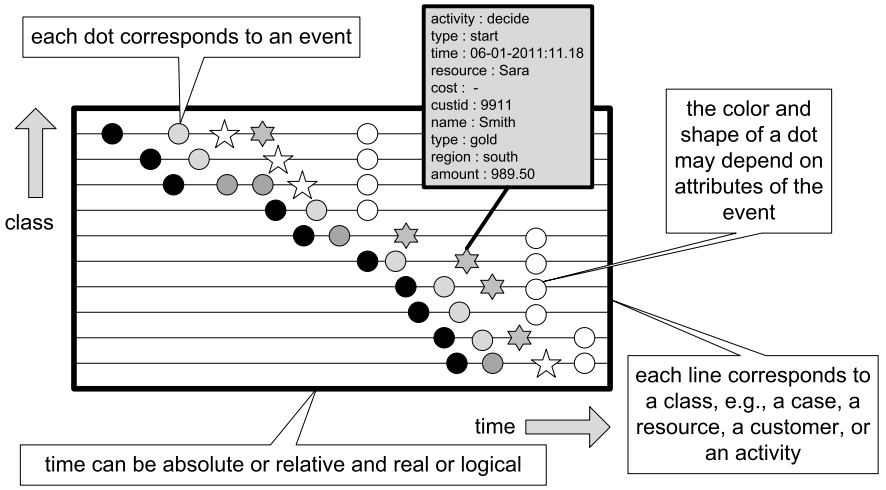

Fig. 9.2 Dotted chart: events are visualized as *dots*. Their position, color, and shape depend on the attributes of the corresponding event

second event has time 1, etc. Also logical time can be absolute (global numbering) or relative (each case starts at time 0).

As Fig. 9.2 shows, the shape and color of a dot may depend on other attributes, i.e., there is also a classifier for the shape of the dot and a classifier for the color of the dot. For instance, if the classifier $\underline{e} = \#_{case}(e)$ is used, then every line corresponds to a case. The shape of the dot may depend on the resource triggering the corresponding event and the color of the dot may depend on the name of the corresponding activity. In our example, the shape of the dot can also depend on the type of customer (silver or gold) and the color of the dot may depend on the region (north, east, south, or west).

Figure 9.2 shows only a schematic view of the dotted chart. Figures 9.3 and 9.4 show two dotted charts based on a real event log. The event log was extracted from the database of a large Dutch housing agency. The event log contains 5987 events relating to 208 cases and 74 activity names. Each case refers to a housing unit (e.g., an apartment). The case starts when the tenant wants to terminate the current lease and ends when the new tenant has moved into the unit. Both figures show 5987 dots. The classifier $\underline{e} = \#_{case}(e)$ is used, i.e., every line corresponds to a unit. The color of the dot depends on the name of the corresponding activity. There are 74 colors: one for each of the possible activities. Figure 9.3 uses absolute/real times for the horizontal dimension. Cases are sorted by the time of the first event. These initial events do not form a straight line. If the arrival rate of new cases would be constant, the frontier formed by initial events would resemble a straight line rather than the curve shown in Fig. 9.3. The curved frontier line shows that the arrival process increases in intensity towards the middle of the time window visualized in Fig. 9.3. Moreover, it seems that events are not evenly spread over the time window. There are periods with little activity. Figure 9.4 uses relative/real times. This figure shows

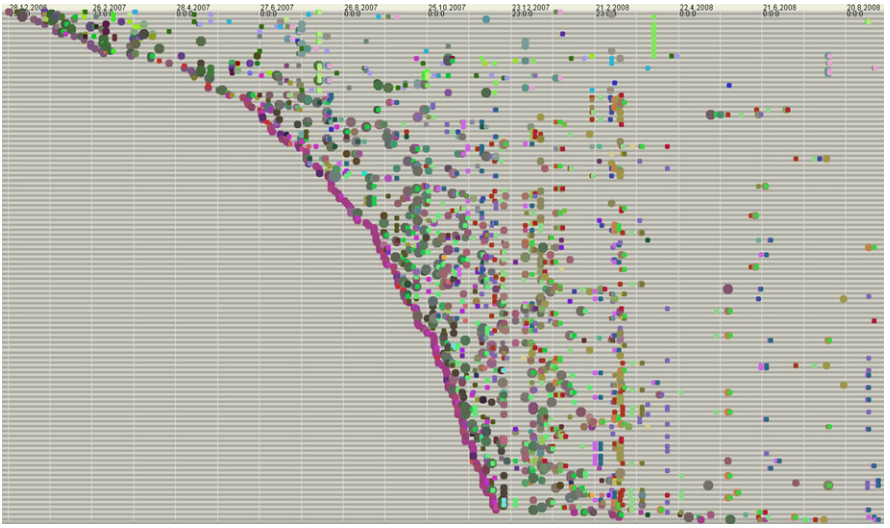

Fig. 9.3 Dotted chart for a process of a housing agency using absolute time. The influx of new cases increases over time. Moreover, several periods with little activity can be identified

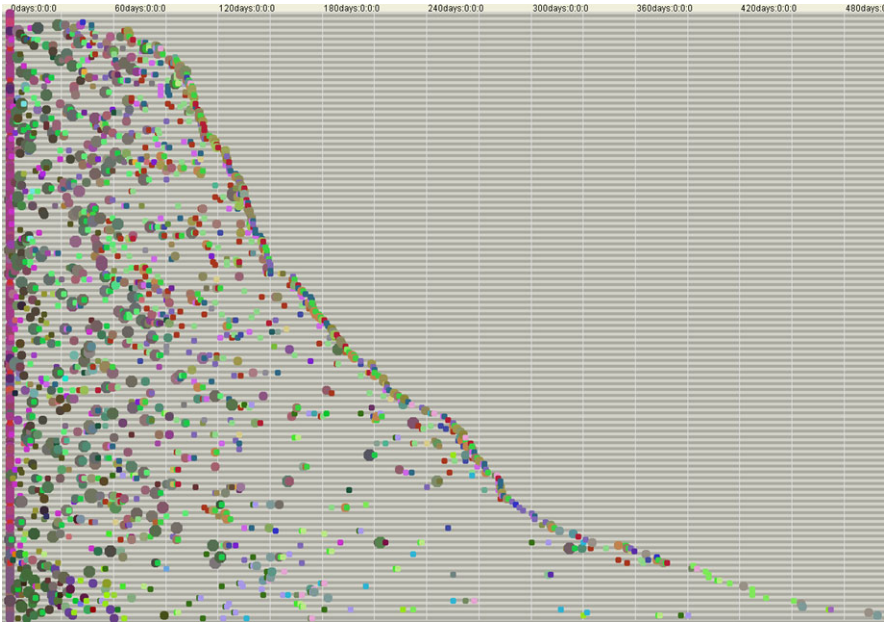

Fig. 9.4 Dotted chart for the process of the housing agency using relative time, i.e., all cases start at time zero. The chart reveals a large variation in flow times: some cases are handled in a few days whereas others take more than a year

that there is a huge variation in flow time. About 45% of the cases are handled in less than 150 days whereas about 10% of the cases take more than one year.

The dotted chart is a very powerful tool to view a process from different angles. One can see all events in one glance while potentially showing different perspectives at the same time (class, color, shape, and time). Moreover, by zooming in one can investigate particular patterns. For example, when classifier $\underline{e} = \#_{resource}(e)$ is used one can immediately see when a resource has been inactive for a longer period.

In the dotted charts shown in this section, timestamps are used to align events in the horizontal dimension. As shown in [79], it is also possible to align events based on their context rather than time. As a result repeating patterns in the event log are aligned so that it becomes easy to see common behavior and deviations without constructing a process model. The identification of such patterns helps understanding the "raw" behavior captured in the event log. As indicated in Chap. 5, event logs often contain low-level events that are of little interest to management. The challenge is to aggregate low-level events into events that are meaningful for stakeholders. Therefore, the event log is often preprocessed after a visual inspection of the log using dotted charts. There are several approaches to preprocess low-level event logs. For example, frequently appearing low-level patterns can be abstracted into events representing activities at the business level [77]. Also activity-based filtering can be used to preprocess the log. We elaborate on this in Chaps. 14 and 15.

The dotted chart can be seen as an example of a *visual analytics* technique. Visual analytics leverages on the remarkable capabilities of humans to visually identify patterns and trends in large datasets. Even though Fig. 9.3 shows almost six thousand events, people involved in this process can see patterns, trends, and irregularities in one glance.

9.3 Organizational Mining

Organizational mining focuses on the *organizational perspective* [130, 159]. Starting point for organizational mining is typically the $\#_{resource}(e)$ attribute present in most event logs. Table 9.3 shows a fragment of a larger event log in which each event has a resource attribute; all complete events have been projected onto their resource and activity attributes. This event log is based on the process model from Chap. 2. Using such information, there are techniques to learn more about people, machines, organizational structures (roles and departments), work distribution, and work patterns.

By analyzing an event log as shown in Table 9.3 it is possible to analyze the relation between resources and activities. Table 9.4 shows the mean number of times a resource performs an activity per case. For instance, activity a is executed exactly once for each case (take the sum of the first column). Pete, Mike, and Ellen are the only ones executing this activity. In 30% of the cases, a is executed by Pete, 50% is executed by Pete, and 20% is executed by Ellen. Activities e and f are always executed by Sara. Activity e is executed, on average, 2.3 times per case. The event

Table 9.3 Compact representation of the event log highlighting the resource attribute of each event (a = *register request*, b = *examine thoroughly*, c = *examine casually*, d = *check ticket*, e = *decide*, f = *reinitiate request*, g = *pay compensation*, and h = *reject request*)

Case id	Trace
1	$\langle a^{Pete}, b^{Sue}, d^{Mike}, e^{Sara}, h^{Pete} \rangle$
2	$\langle a^{Mike}, d^{Mike}, c^{Pete}, e^{Sara}, g^{Ellen} \rangle$
3	$\langle a^{Pete}, c^{Mike}, d^{Ellen}, e^{Sara}, f^{Sara}, b^{Sean}, d^{Pete}, e^{Sara}, g^{Ellen} \rangle$
4	$\langle a^{Pete}, d^{Mike}, b^{Sean}, e^{Sara}, h^{Ellen} \rangle$
5	$\langle a^{Ellen}, c^{Mike}, d^{Pete}, e^{Sara}, f^{Sara}, d^{Ellen}, c^{Mike}, e^{Sara}, f^{Sara}, b^{Sue}, d^{Pete}, e^{Sara}, h^{Mike} \rangle$
6	$\langle a^{Mike}, c^{Ellen}, d^{Mike}, e^{Sara}, g^{Mike} \rangle$
...	...

Table 9.4 Resource-activity matrix showing the mean number of times a person performed an activity per case

	a	b	c	d	e	f	g	h
Pete	0.3	0	0.345	0.69	0	0	0.135	0.165
Mike	0.5	0	0.575	1.15	0	0	0.225	0.275
Ellen	0.2	0	0.23	0.46	0	0	0.09	0.11
Sue	0	0.46	0	0	0	0	0	0
Sean	0	0.69	0	0	0	0	0	0
Sara	0	0	0	0	2.3	1.3	0	0

log conforms to the process model shown in Fig. 2.2. Hence, for some cases e is executed only once whereas for other cases e is executed repeatedly (2.3 times on average). On average, activity f is executed 1.3 times. This suggests that the middle part of the process (composed of activities b, c, d, e, and f) needs to be redone for the majority of cases. Consider for example case 5 in Table 9.3; e is executed three times and f is executed twice for this case.

9.3.1 Social Network Analysis

Sociometry, also referred to as sociography, refers to methods that present data on interpersonal relationships in graph or matrix form [182]. The term sociometry was coined by Jacob Levy Moreno who already used such techniques in the 1930s to better assign students to residential cottages in a training facility. Until recently, the input data for sociometry consisted mainly of interviews and questionnaires. However, with the availability of vast amounts of electronic data, new ways of gathering input data are possible.

Here we restrict ourselves to *social networks* as shown in Fig. 9.5. The nodes in a social network correspond to organizational entities. Often, but not always, there is a one-to-one correspondence between the resources found in the log and organizational entities (i.e., nodes). In Fig. 9.5 nodes x, y, and z could refer to persons.

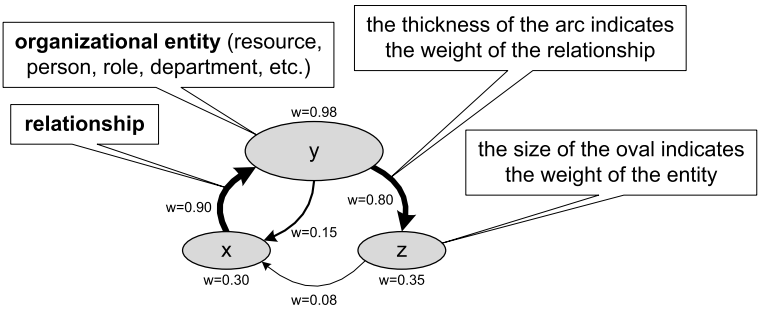

Fig. 9.5 A social network consists of nodes representing organizational entities and arcs representing relationships. Both nodes and arcs can have weights indicated by "$w = \cdots$" and the size of the shape

The nodes in a social network may also correspond to aggregate organizational entities such as roles, groups, and departments. The arcs in a social network correspond to relationships between such organizational entities. Arcs and nodes may have *weights*. The weight of an arc or node indicates its *importance*. For instance, node y is more important than x and z as is indicated by its size. The relationship between x and y is much stronger than the relationship between z and x as shown by the thickness of the arc. The interpretation of "importance" depends on the social network. Later, we will give some examples to illustrate the concept.

Sometimes the term *distance* is used to refer to the inverse of the weight of an arc. An arc connecting two organizational entities has a high weight if the distance between both entities is small. If the distance from node x to node y is large, then the weight of the corresponding arc is small (or the arc is not present in the social network).

A wide variety of metrics have been defined to analyze social networks and to characterize the role of individual nodes in such a diagram [182]. For example, if all other nodes are in short distance to a given node and all geodesic paths (i.e., shortest paths in the graph) visit this node, then clearly the node is very central (like a spider in the web). There are different metrics for this intuitive notion of *centrality*. The Bavelas–Leavitt index of centrality is a well-known example that is based on the geodesic paths in the graph. Let i be an node and let $D_{j,k}$ be the geodesic distance from node j to node k. The Bavelas–Leavitt index of centrality is defined as $BL(i) = (\sum_{j,k} D_{j,k})/(\sum_{j,k} D_{j,i} + D_{i,k})$. The index divides the sum of all geodesic distances by the sum of all geodesic distances from and to node i. Other related metrics are *closeness* (1 divided by the sum of all geodesic distances to a given node) and *betweenness* (a ratio based on the number of geodesic paths visiting a given node) [159, 182]. Recall that distance can be seen as the inverse of arc weight.

Notions such as centrality analyze the position of one organizational entity, say a person, in the whole social network. There are also metrics making statements about the network as a whole, e.g., the degree of connectedness. Moreover, there are also

Table 9.5 Handover of work matrix showing the mean number of handovers from one person to another per case

	Pete	Mike	Ellen	Sue	Sean	Sara
Pete	0.135	0.225	0.09	0.06	0.09	1.035
Mike	0.225	0.375	0.15	0.1	0.15	1.725
Ellen	0.09	0.15	0.06	0.04	0.06	0.69
Sue	0	0	0	0	0	0.46
Sean	0	0	0	0	0	0.69
Sara	0.885	1.475	0.59	0.26	0.39	1.3

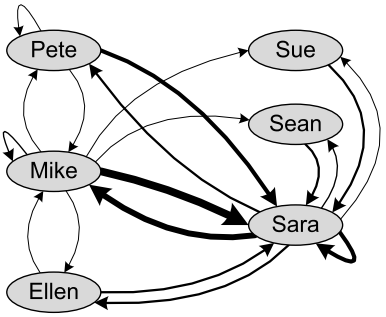

Fig. 9.6 Social network based on handover of work at the level of individual resources using a threshold of 0.1. The thickness of the arcs is based on the frequency of handovers from one person to another

techniques to identify *cliques* (groups of entities that are strongly connected to each other while having fewer connections to entities outside the clique).

Clearly event logs with $\#_{resource}(e)$ attributes provide an excellent source of information for social network analysis. For instance, based on the event log one can count the number of times *work is handed over from one resource to another*. Consider for example case 1 having the following trace: $\langle a^{Pete}, b^{Sue}, d^{Mike}, e^{Sara}, h^{Pete} \rangle$. Clearly, there is a handover of work from Pete to Sue and Mike after the completion of a. Note that Sue does not hand over work to Mike, because b and d are concurrent. However, both Sue and Mike hand over work to Sara, because activity e requires input from both b and d. Finally, Sara hands over work to Pete. Hence, in total there are five handovers: (a^{Pete}, b^{Sue}), (a^{Pete}, d^{Mike}), (b^{Sue}, e^{Sara}), (d^{Mike}, e^{Sara}), and (e^{Sara}, h^{Pete}). Table 9.5 shows the average number of handovers from one resource to another. For instance, Mike frequently hands over work to Sara: on average 1.725 times per case. Sue and Sean only hand over work to Sara as they only execute activity b. It is important to note that the discovered process model is exploited when constructing the social network. The causal dependencies in the process model are used to count handovers in the event log. This way only "real" handovers of work are counted, e.g., concurrent activities may follow one another but do not contribute the number of handovers.

Table 9.5 encodes a social network. All non-zero cells represent "handover of work" relationships. When visualizing a social network typically a threshold is used. If we set the threshold to 0.1, we obtain the social network shown in Fig. 9.6. All cells with a value of at least 0.1 are turned into arcs in the social network. To keep the diagram simple, we only assigned weights to arcs and not to nodes. As Fig. 9.6

Table 9.6 Handover of work matrix at the role level

	Assistant	Expert	Manager
Assistant	1.5	0.5	3.45
Expert	0	0	1.15
Manager	2.95	0.65	1.3

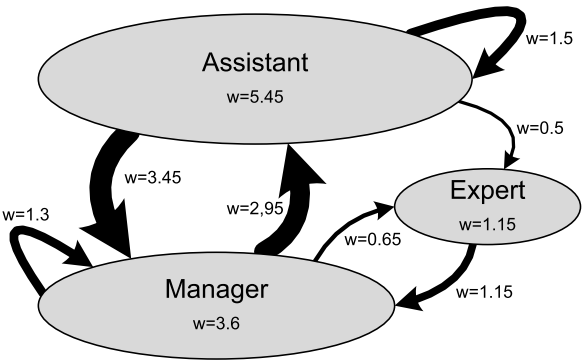

Fig. 9.7 Social network based on handover of work at the level of roles. The weights of nodes are based on the number of times a resource having the role performs an activity. The weights of the arcs are based of the average number of times a handover takes place from one role to another per case

shows there is a strong connection between Mike and Sara. On average, there are 1.725 handovers from Mike to Sara and 1.475 handovers from Sara to Mike. The social network clearly shows the flow of work in the organization and can be used to compute metrics such as the Bavelas–Leavitt index of centrality. Such analysis shows that Sara and Mike are most central in the social network.

The nodes in a social network correspond to organizational entities. In Fig. 9.6 the entities are individual resources. However, it is also possible to construct social networks at the level of departments, teams, or roles. Assume for example that there are three roles: *Assistant*, *Expert*, and *Manager*. Pete, Mike, and Ellen have the role *Assistant*, Sue and Sean have the role *Expert*, and Sara is the only one having the role *Manager*. Later, we will show that such roles can be discovered from frequent patterns in the event log. Moreover, such information is typically available in the information system. Now we can count the number of handovers at the role level. Consider again case 1: $\langle a^{Pete}, b^{Sue}, d^{Mike}, e^{Sara}, h^{Pete}\rangle$. Using the information about roles we can rewrite this trace to $\langle a^{Assistant}, b^{Expert}, d^{Assistant}, e^{Manager}, h^{Assistant}\rangle$. Again we find five handovers: one from role *Assistant* to role *Expert* ($a^{Assistant}, b^{Expert}$), one from role *Assistant* to role *Assistant* ($a^{Assistant}, d^{Assistant}$), one from role *Expert* to role *Manager* ($b^{Expert}, e^{Manager}$), one from role *Assistant* to role *Manager* ($d^{Assistant}, e^{Manager}$), and one from role *Manager* to role *Assistant* ($e^{Manager}, h^{Assistant}$). Table 9.6 shows the average frequency of such handovers per case. This matrix containing sociometric information can be converted into a social network as shown in Fig. 9.7.

The social network in Fig. 9.7 has weighted nodes and arcs. The weights are visualized graphically. For instance, the biggest node is role *Assistant* with a weight of 5.45. This weight indicates the average number of activities executed by this role.

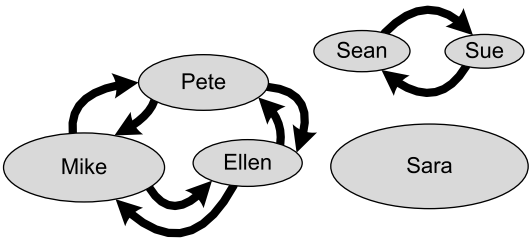

Fig. 9.8 Social network based on similarity of profiles. Resources that execute similar collections of activities are related. Sara is the only resource executing e and f. Therefore, she is not connected to other resources. Self-loops are suppressed as they contain no information (self-similarity)

The weight of role *Expert* is only 1.15, because the two experts (Sue and Sean) only execute activity b which, on average, is executed 1.15 times per case. The weights of the arcs are directly taken from Table 9.6. Clearly, handovers among the roles *Assistant* and *Manager* are most frequent.

Counting handovers of work is just one of many ways of constructing a social network from an event log. In [159] various types of social networks are presented. For example, one can simply count how many times two resources have worked on the same case, i.e., two nodes have a strong relationship when they frequently work together on common cases. One can also use Table 9.4 to quantify the *similarity* of two resources. Every row in the resource-activity matrix can be seen as the *profile* of a resource. Such a vector describes the relevant features of a resource. For example, Pete has profile $P_{Pete} = (0.30, 0.0, 0.345, 0.69, 0.0, 0.0, 0.135, 0.165)$, Mike has profile $P_{Mike} = (0.5, 0.0, 0.575, 1.15, 0.0, 0.0, 0.225, 0.275)$, and Sara has profile $P_{Sara} = (0.0, 0.0, 0.0, 0.0, 2.3, 1.3, 0.0, 0.0)$. Clearly, P_{Pete} and P_{Mike} are very similar whereas P_{Pete} and P_{Sara} are not. The distance between two profiles can be quantified using well-known distance measures such as the *Minkowski distance*, *Hamming distance*, and *Pearson's correlation coefficient*. Moreover, clustering techniques such as *k-means clustering* and *agglomerative hierarchical clustering* can be used to group similar resources together based on their profile (see Sect. 4.3). Two resources in the same cluster (or in close proximity according to the distance metric) are strongly related whereas resources in different clusters (or far away from each other) have no significant relationship in the social network.

For the resource-activity matrix shown in Table 9.4 it does not matter which distance metric or clustering technique is used. All will come to the conclusion that Pete, Mike, and Ellen are very similar and thus have a strong relationship in the social network based on similarity. Similarly, Sue and Sean have a strong relationship in the social network based on similarity. Sara is clearly different from the resources in the two other groups. Figure 9.8 shows the social network based on similarity. Here one can clearly see the roles *Assistant*, *Expert*, and *Manager* mentioned before. However, now the roles are discovered based on the profiles of the resources.

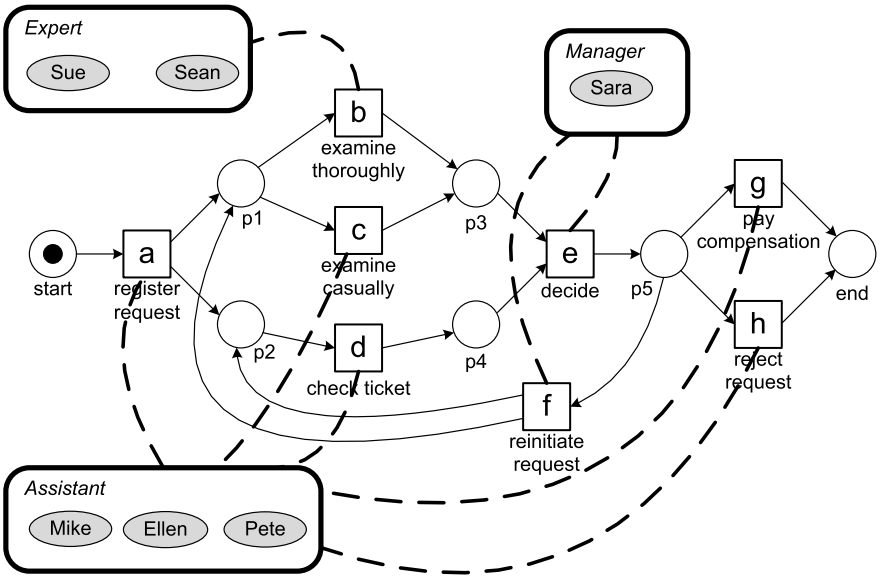

Fig. 9.9 Organizational model discovered based on the event log

9.3.2 Discovering Organizational Structures

The behavior of a resource can be characterized by a *profile*, i.e., a vector indicating how frequent each activity has been executed by the resource. By using such profiles, various clustering techniques can be used to discover similar resources. Figure 9.8 showed an example in which three roles are discovered based on similarities of the profiles of the six resources. In Sect. 4.3 we introduced k-means clustering and agglomerative hierarchical clustering. For k-means clustering the number of clusters is decided upfront. Agglomerative hierarchical clustering produces a dendrogram allowing for a variable number of clusters depending on the desired granularity. Additional relevant features of resources (authorizations, salary, age, etc.) can be added to the profile before clustering. This all depends on the information available. After clustering the resources into groups, these groups can be related to activities in the process. Figure 9.9 shows the end result using the roles discovered earlier.

The three roles *Assistant*, *Expert*, and *Manager* in Fig. 9.9 have the property that they partition the set of resources. In general this will not be the case, e.g., a resource can have multiple roles (e.g., a consultant that is also team leader). Moreover, each activity corresponds to precisely one role. Also this does not always need to be the case. Figure 9.10 sketches a more general situation.

The hypothetical organizational model in Fig. 9.10 connects the process model and the resources seen in the event log. There are eighth organizational entities: $oe1, \ldots, oe8$. The model is hierarchical, e.g., $oe4$ contains resource $r5$ and all resources of $oe6$, $oe7$, and $oe8$. Hence five resources belong to organizational entity

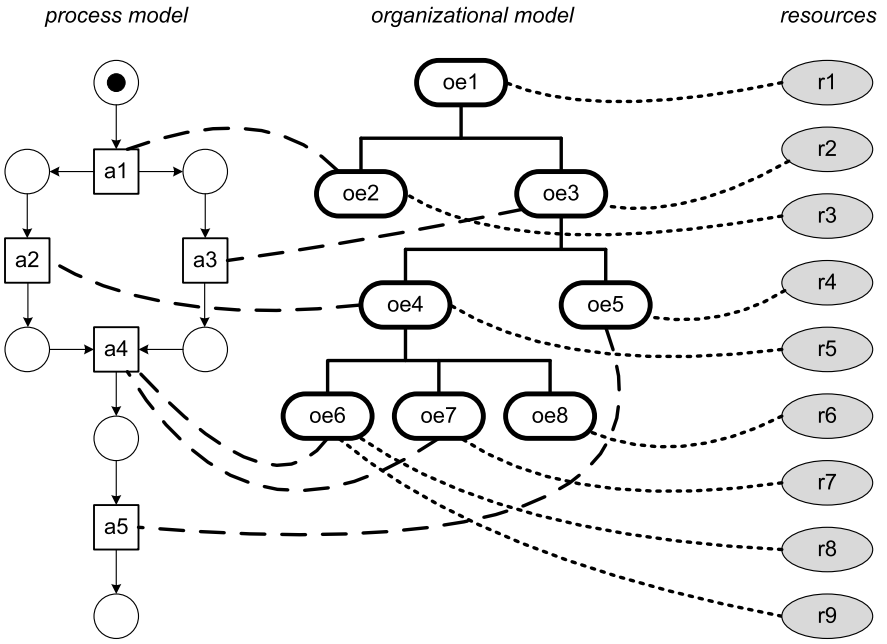

Fig. 9.10 The organizational entities discovered connect activities in the process model to sets resources

$oe4$: $r5$, $r6$, $r7$, $r8$, and $r9$. Organizational entity $oe1$, i.e., the root node, contains all nine resources. If agglomerative hierarchical clustering is used to cluster resources, one automatically gets such a hierarchical structure. Figure 4.7 in Sect. 4.3 shows how agglomerative hierarchical clustering creates a *dendrogram* and Fig. 4.8 shows how any horizontal line defines a level in the hierarchy. The translation from a dendrogram to a hierarchical structure as shown in Fig. 9.10 is straightforward.

Activity $a1$ in Fig. 9.10 can only be performed by resource $r3$ whereas activity $a2$ can be executed by $r5$, $r6$, $r7$, $r8$, or $r9$. For more information about organizational mining we refer to [130].

9.3.3 Analyzing Resource Behavior

Figure 9.10 shows how activities, organizational entities, and resources can be related. Since events in the log refer to activities and resources (and indirectly also to organizational entities), performance measures extracted from the event log can be projected onto such models. For instance, frequencies can be projected onto activities, organizational entities, and resources. It could be shown that resource $r5$ performed 150 activities in the last month: 100 times $a2$ and 50 times $a3$. By aggregating such information it could be deduced that organizational entity $oe4$ was used 300 times in the same period.

In Table 9.3, we abstracted from transaction types, i.e., we did not consider the start and completion of an activity instance. Most logs will contain such information. For example, Table 9.1 shows the start and completion of each activity instance. Some logs will even show when a workitem is offered to a resource or when it is assigned. If such events are recorded, then a diagram such as Fig. 9.10 can also show detailed time related information. For example, the utilization and response times of resources can be shown.

Assuming that the event log contains high quality information including precise timestamps and transaction types, the behavior of resources can be analyzed in detail [139]. Of course privacy issues play an important role here. However, the event log can be anonymized prior to analysis. Moreover, in most organizations one would like to do such analysis at an aggregate level rather than at the level of individuals. For instance, in Sect. 3.1, we mentioned the Yerkes-Dodson law of arousal which describes the relation between workload and performance of people. This law hypothesizes that people work faster when the workload increases. If the event log contains precise timestamps and transaction types, then it is easy to empirically investigate this phenomenon. For any activity instance, one knows its duration and by scanning the log it is also easy to see what the workload was when the activity instance was being performed by some resource. Using supervised learning (e.g., regression analysis or decision tree analysis) the effects of different workloads on service and response times can be measured. See [139] for more examples.

Privacy and anonymization

Event logs may contain sensitive or private data. Events refer to actions and properties of customers, employees, etc. For instance, when applying process mining in a hospital it is important to ensure data privacy. It would be unacceptable that data about patients would be used by unauthorized persons or that event data about treatments would be used in a way not intended when releasing the data. The challenge in process mining is to use event logs to improve processes and information systems while protecting personally identifiable information and not revealing sensitive data. Therefore, most event logs contain *anonymized* attribute values. For example, the name of the customer or employee is often irrelevant for questions that need to be answered. To make an attribute anonymous, the original value is mapped onto a new value in a deterministic manner. This ensures that one can correlate attributes in one event to attributes in another event without knowing the actual values. For instance, all occurrences of the name "Wil van der Aalst" are mapped onto "Q2T4R5R7X1Y9Z". The mapping of the original value onto the anonymized value should be such that it is not easy (or even impossible) to compute the inverse of the mapping. Anonymous data can sometimes be de-anonymized by combining different data sources. For example, it is often possible to trace back an individual based on her birth date and the birth dates of her children. Therefore, even "anonymous data" should be handled carefully.

Table 9.7 Compact representation of the event log highlighting timestamps; artificial timestamps are used to simplify the presentation of the time-based replay approach

Case id	Trace
1	$\langle a^{12}_{start}, a^{19}_{complete}, b^{25}_{start}, d^{26}_{start}, b^{32}_{complete}, d^{33}_{complete}, e^{35}_{start}, e^{40}_{complete}, h^{50}_{start}, h^{54}_{complete}\rangle$
2	$\langle a^{17}_{start}, a^{23}_{complete}, d^{28}_{start}, c^{30}_{start}, d^{32}_{complete}, c^{38}_{complete}, e^{50}_{start}, e^{59}_{complete}, g^{70}_{start}, g^{73}_{complete}\rangle$
3	$\langle a^{25}_{start}, a^{30}_{complete}, c^{32}_{start}, c^{35}_{complete}, d^{35}_{start}, d^{40}_{complete}, e^{45}_{start}, e^{50}_{complete}, f^{50}_{start}, f^{55}_{complete}, b^{60}_{start}, d^{62}_{start}, b^{65}_{complete}, d^{67}_{complete}, e^{80}_{start}, e^{87}_{complete}, g^{90}_{start}, g^{98}_{complete}\rangle$
...	...

Note that process mining techniques do not create *new* data. The information stored in event logs originates from other databases and audit trails. Therefore, privacy and security issues already exist before applying process mining. Nevertheless, the active use of data and process mining techniques increases the risk of data misuse. Organizations should therefore continuously balance the benefits of creating and using event data against potential privacy and security problems.

9.4 Time and Probabilities

The *time perspective* is concerned with the timing and frequency of events. In most event logs, events have a timestamp ($\#_{time}(e)$). The granularity of these timestamps may vary. In some logs only date information is given, e.g., "30-12-2010". Other event logs have timestamps with millisecond precision. The presence of timestamps enables the discovery of bottlenecks, the analysis of service levels, the monitoring of resource utilization, and the prediction of remaining processing times of running cases. In this section we focus on *replaying event logs with timestamps*. A small modification of the replay approach presented in Sect. 8.2 suffices to include the time perspective in process models.

Table 9.7 shows a fragment of some larger event log highlighting the role of timestamps. To simplify the presentation, we use fictive two-digit timestamps rather than verbose timestamps like "30-12-2010:11.02". Moreover, we assume that each event has a start event and a complete event. Obviously, the replay approach does not depend on these simplifying assumptions.

Figure 9.11 shows some raw diagnostic information after replaying the three cases shown in Table 9.7. Activity a has three activity instances; one for each case. The first instance of a runs from time 12 to time 19. Hence, the duration of this activity instance is 7 time units. Activity d has four activity instances. For case 3 there are two instances of d; one running from time 35 to time 40 and one running from time 62 to time 67. The durations of all activity instances are shown. Also places are annotated to indicate how long tokens remained there. For example, there

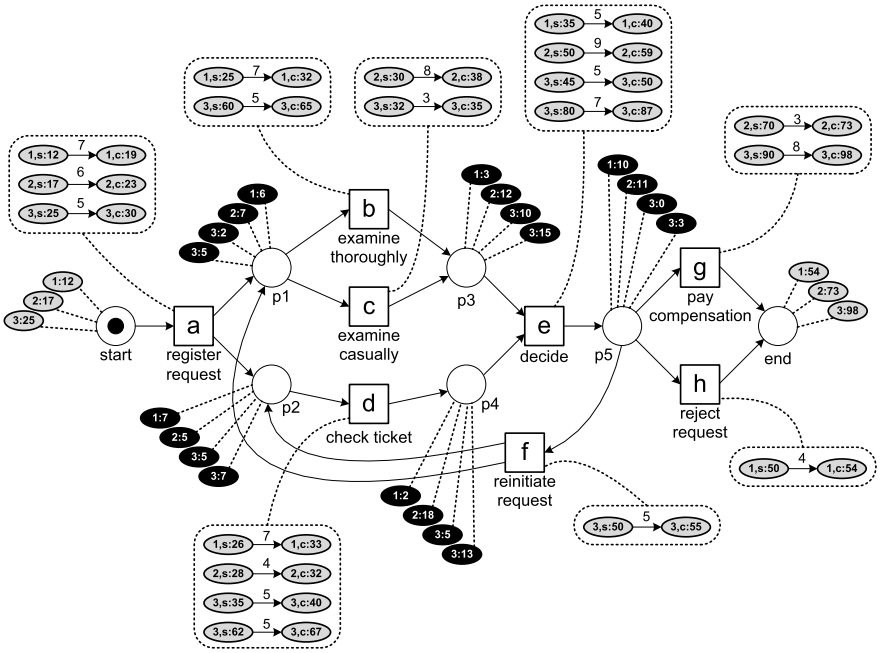

Fig. 9.11 Timed replay of the first three cases in the event log: case 1 starts at time 12 and ends at time 54, case 2 starts at time 17 and ends at time 73, case 3 starts at time 25 and ends at time 98

were four periods in which a token resided in place $p1$: one token corresponding to case 1 resided in $p1$ for 6 time units (from time 19 until time 25), one token corresponding to case 2 resided in $p1$ for 7 time units (from time 23 until time 30), and two tokens corresponding to case 3 resided in this place (one for $32 - 30 = 2$ time units and one for $60 - 55 = 5$ time units). These times can be found using the approach presented in Sect. 8.2. The only modifications are that now tokens bear timestamps and statistics are collected during replay. In this example, all three cases fit perfectly (i.e., no missing or remaining tokens). One needs to ignore non-fitting events or cases to deal with logs that do not have a conformance of 100%. Heuristics are needed to deal with such situations, but here we assume perfect fitness.

Figure 9.12 shows another view on the information gathered while replaying the three cases. Consider for instance case 3. For this case, an instance of activity a was running from time 25 until time 30. At time 30, c and d became enabled. However, as shown, c started at time 32 and d started at time 35. This implies that there was a waiting time of 2 before c started and a waiting time of 5 before d started. After completing c and d, i.e., at time 40, the first instance of e became enabled. Since the first instance of activity e ran from time 45 until 50, the waiting time for this instance of e was $45 - 40 = 5$ time units. Note that from time 35 until time 45 there was a token in place $p3$ (because c completed at time 35 and e started at time 45). However, only half of this period should be considered as waiting time for e, because e only got enabled at time 40 when d completed. As discussed in Sect. 8.5.3, such

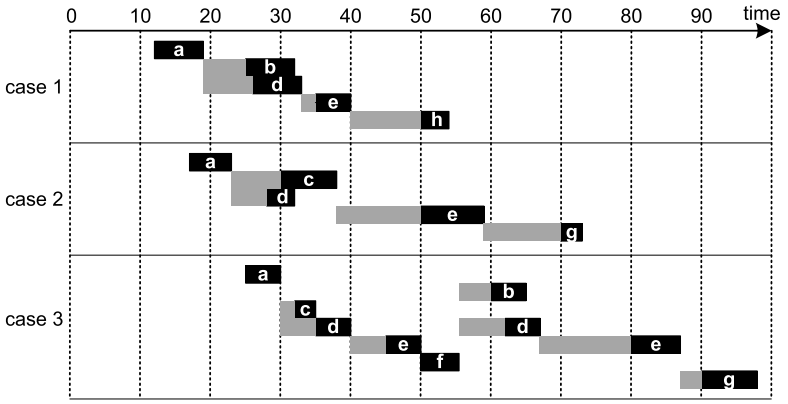

Fig. 9.12 Timeline showing the activity instances of the first three activities

diagnostics are *only possible because events in the log have been coupled to model elements through replay*.

After replay, for each place a collection of "token visits" has been recorded. Each token visit has a start and end time. Hence, a multi-set of durations can be derived. In the example, place $p1$ has the multi-set $[6, 7, 2, 5, \ldots]$ of durations. For a large event log such a multi-set will contain thousands of elements. Hence, it is possible to fit a distribution and to compute standard statistics such as mean, standard deviation, minimum, and maximum. The same holds for activity instances. Every activity instance has a start and end time. Hence, a multi-set of service times can be derived. For example, activity e in the example has the multi-set $[5, 9, 5, 7, \ldots]$ of activity durations. Also here standard statistics can be computed. These can also be computed for waiting times. It is also possible to compute confidence intervals to derive statements such as "the 90% confidence interval for the mean waiting time for activity x is between 40 and 50 minutes".

Figures 9.11 and 9.12 demonstrate that replay can be used to provide various kinds of performance related information:

- *Visualization of waiting and service times.* Statistics such as the average waiting time for an activity can be projected onto the process model. Activities with a high variation in service time could be highlighted in the model, etc.
- *Bottleneck detection and analysis.* The multi-set of durations attached to each place can be used to discover and analyze bottlenecks. The places where most time is spent can be highlighted. Moreover, cases that spend a long time in a particular place can be further investigated. This is similar to the selection of non-conforming cases described earlier (cf. Fig. 8.8), i.e., the sublog of delayed cases can be analyzed separately to find root causes for the delays.
- *Flow time and SLA analysis.* Fig. 9.11 also shows that the overall flow time can be computed. (In fact, no process model is needed for this.) One can also point to two arbitrary points in the process, say x and y, and compute how many times

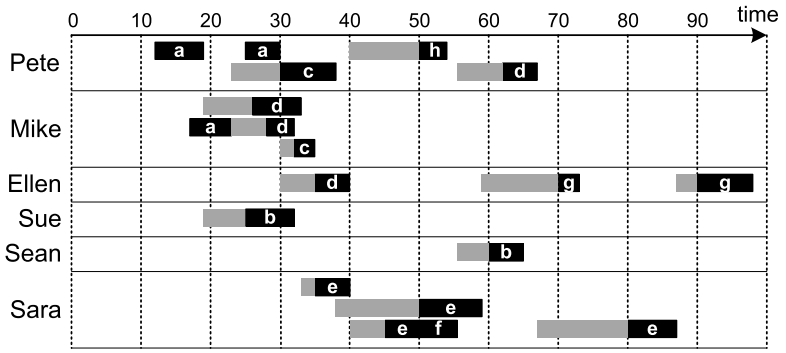

Fig. 9.13 Timeline showing the activity instances projected onto resources. Such projections of the event log allow for the analysis of resource behavior and their utilization

a case flows from x to y. The multi-set of durations to go from x to y can be used to compute all kinds of statistics, e.g., the average flow time between x and y or the fraction of cases taking more than some preset norm. This can be used to monitor Service Level Agreements (SLAs). For instance, it could be that there is a contractual agreement that for 90% of the cases y should be executed within 48 hours after the completion of x. Non-conformance with respect to such an SLA can be highlighted in the model.

- *Analysis of frequencies and utilization.* While replaying the model, times and frequencies are collected. These can be used to show routing probabilities in the model. For example, after e there is a choice to do f, g or h. By analyzing frequencies, one can indicate in the model that in 56% of choices, e is followed by f, in 20% g is chosen, and in 24% h is chosen. By combining frequencies and average service times one can also compute the utilization of resources. Figure 9.13 shows all activity instances and their waiting times *projected onto the resources* executing them. This illustrates that replay can be used to analyze resource behavior.

The event log shown in Table 9.7 only contains *start* and *complete* events. In Chap. 5 we identified additional event types such as *assign*, *schedule*, *suspend*, *resume*, *manualskip*, *abort_case*, and *withdraw*. If an event log contains such events, more statistics can be collected during replay. For instance, if *start* events are preceded by *assign* events, it is possible to analyze how long it takes to start executing the activity instance after being assigned to a specific resource. The transactional life-cycle shown in Fig. 5.3 can be used when replaying the event log.

It can also be the case that the event log does not contain transactional information, i.e., the $\#_{trans}(e)$ attribute is missing in the log. In this case activities are assumed to be atomic. Nevertheless, it is still possible to analyze the time that passes in-between such atomic activities. In addition, heuristics can be applied to "guess" the duration of activity instances.

9.5 Decision Mining

The *case perspective* focuses on properties of cases. Each case is characterized by its case attributes, the attributes of its events, the path taken, and performance information (e.g., flow times).

First, we focus on the influence of case and event attributes on the routing of cases. In Fig. 9.9 there are two *decision points*:

- after registering the request (activity a) either a thorough examination (activity b) or a casual examination (activity c) follows; and
- after making a decision (activity e), activity g (pay compensation), activity h (reject request), or activity f (reinitiate request) follows.

Both decision points are of type XOR-split: precisely one of several alternatives is chosen. *Decision mining* aims to find rules explaining such choices in terms of characteristics of the case [120]. For example, by analyzing the event log used to discover Fig. 9.9 one could find that customers from the southern region are always checked thoroughly and that requests by silver customers always get rejected. Clearly, *a classification technique like decision tree learning can be used to find such rules* (see Sect. 4.2). Recall that the input for decision tree learning is a table where every row lists one categorical response variable (e.g., the chosen activity) and multiple predictor variables (e.g., properties of the customer). The decision tree aims to explain the response variable in terms of the predictor variables.

Consider, for example, the situation shown in Fig. 9.14. Using three different notations (YAWL, BPMN, and Petri nets) a choice is depicted: activity x is followed by either activity y or activity z. The table in Fig. 9.14 shows different cases for which this choice needs to be made. There are three predictor variables (*type*, *region*, and *amount*) and one response variable (*activity*). Variables *type*, *region*, and *activity* are categorical and variable *amount* is numerical. The predictor variables correspond to knowledge known about the case at the point in time when the decision was made. The response variable *activity* is determined based on a scan of the event log. The event log will reveal whether x was followed by y or z. The table in Fig. 9.14 serves as input for some decision tree learning algorithm as explained in Sect. 4.2. The resulting decision tree can be rewritten into a rule. Based on the example table, classification will show that the value of the response variable is y if the customer is a gold customer and the amount is lower than € 500. Otherwise, the value of the response variable is z as shown in Fig. 9.14.

Petri nets cannot express OR-splits and joins directly. However, in higher-level languages like BPMN and YAWL one can express such behavior. Figure 9.15 shows an OR-split using the YAWL and BPMN notation: activity x is followed by y, or z, or y and z. Note that the response variable *activity* is still categorical and can be determined by scanning the log. The table in Fig. 9.15 can be analyzed using a decision tree learner and the result can be transformed into one rule for each of the output arcs. The response variable is "just y" if the customer is a gold customer and the amount is less than € 500, the response variable is "just z" if the customer is a silver customer and the amount is at least € 500, and the response variable is

type	region	amount	activity
gold	south	987.30	z
silver	north	178.70	z
gold	south	211.50	y
silver	west	587.70	z
silver	east	224.70	z
silver	south	278.50	z
gold	north	488.50	y
silver	west	443.20	z
silver	south	673.70	z
gold	west	413.50	y
silver	south	687.70	z
gold	south	987.30	z
silver	north	378.80	z
gold	south	314.50	y
silver	north	537.70	z
silver	west	158.70	z
gold	east	344.50	y
...	...	...	...

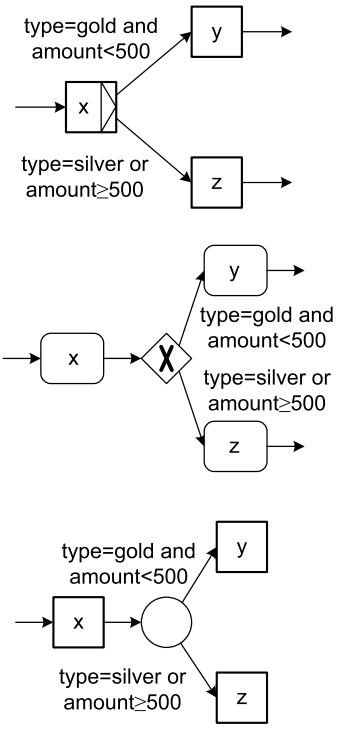

Fig. 9.14 Decision mining: using case and event attributes, a rule is learned for the XOR-split. The result is shown using different notations: YAWL (*top*), BPMN (*middle*), and Petri nets (*bottom*)

type	region	amount	activity
gold	south	987.30	y and z
silver	north	178.70	y and z
gold	south	211.50	just y
silver	west	587.70	just z
silver	east	224.70	y and z
silver	south	278.50	y and z
gold	north	488.50	just y
silver	west	443.20	y and z
silver	south	673.70	just z
...	...	...	...

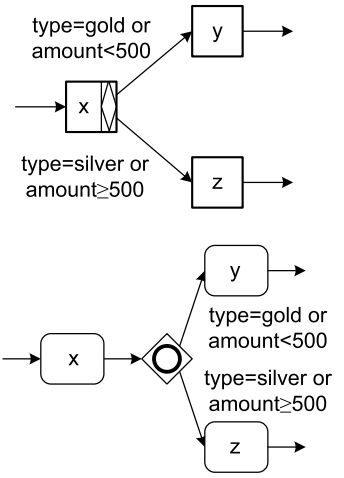

Fig. 9.15 Decision mining: using case and event attributes, a rule is learned for the OR-split. The response variable has three possible values: just y, just z, and both y and z

"*y* and *z*" in all other cases. Based on this classification the conditions shown in the YAWL and BPMN models can be derived.

For the predictor variables all case and event attributes can be used. Consider for instance the decision point following activity *e* and event 35654431 in Table 9.1. The case and event attributes of this event are shown in Tables 9.1 and 9.2. Hence, predictor variables for event 35654431 are: *case* = 1, *activity* = *decide*, *time* = *06-01-2011:11.22*, *resource* = *Sara*, *trans* = *complete*, *cost* = 200, *custid* = 9911, *name* = *Smith*, *type* = *gold*, *region* = *south*, and *amount* = 989.50. As described in [120] also the attributes of earlier and later events can be taken into account. For example, all attributes of all events in the trace up to the decision moment can be used. In the process shown in Fig. 9.9 one could find the rule that all cases that involve Sean get rejected in the decision point following activity *e*.

There may be loops in the model. Hence, the same decision point may be visited multiple times for the same case. Each visit corresponds to a new row in the table used by the decision tree algorithm. For example, in the process shown in Fig. 9.9, there may be cases for which *e* is executed four times. The first three times *e* is be followed by *f* and the fourth time *e* is followed by *g* or *h*. Each of the four decisions corresponds to a row in the table used for classification. Using replay, the outcome of the decision (i.e., the response variable) can be identified for each row. Also note that the values of the predictor variables for these four rows may be different.

In some cases, it may be impossible to derive a reasonable decision rule. The reason may be that there is too little data or that decisions are seemingly random or based on considerations not in the event log. In such cases, replay can be used to provide a probability for each branch. Hence, such a decision point is characterized by probabilities rather than data dependent decision rules.

The procedure can be repeated for all decision points in a process model. The results can be used to extend the process model, thus incorporating the case perspective.

Classification in process mining

The application of classification techniques like decision tree learning is *not limited to decision mining* as illustrated by Figs. 9.14 and 9.15: *additional predictor variables* may be used and *alternative response variables* can be analyzed.

In Figs. 9.14 and 9.15 only attributes of events and cases are used as predictor variables. However, also behavioral information can be used. For instance, in Fig. 9.9 it would be interesting to count the number of times that *f* has been executed. This may influence the decision point following activity *e*. For example, it could be the case that a request is never initiated more than two times. It may also be that timing information is used as a predictor variable. For instance, if the time taken to check the ticket is less than five minutes, then it is more likely that the request is rejected. It is also possible to use *contextual* information as a predictor variable. Contextual information is in-

formation that is not in the event log and that is not necessarily related to a particular case. For example, the weather may influence a decision. This can only be discovered if the weather condition is taken into account as a predictor variable. Decisions may also depend on the volume of work in the pipeline. One can imagine that the choice between *b* and *c* in Fig. 9.9 depends on the workload of the two experts Sue and Sean. When they are overloaded, it may be less likely that *b* is selected. These examples illustrate that predictor variables are not limited to case and event attributes. However, note the "curse of dimensionality" discussed in Sect. 4.6.3. Analyzing decision points with many predictor variables may be computationally intractable.

In Figs. 9.14 and 9.15 we used classification to learn decision rules. The predictor variables can also be used to learn *other properties of the process*. For instance, one may be interested in characterizing cases for which a particular activity is executed. Classification can also be used to uncover reasons for non-conformance. As shown in Fig. 8.8, the event log can be split into two sublogs: one event log containing only fitting cases and one event log containing only non-fitting cases. The observation whether a case fits or not, can be seen as a response variable. Hence, classification techniques like decision tree learning can be used to characterize cases that deviate. For example, one could learn the rule that cases of gold customers from the southern region tend to deviate from the normative model. Similarly, one could learn rules related to the lateness of cases. For instance, one could find out that cases involving Ellen tend to be delayed.

These examples show that established classification techniques can be combined with process mining once the process model and the event log are connected through replay techniques.

9.6 Bringing It All Together

In this chapter we showed that a control-flow model can be extended with additional perspectives extracted from the event log. Figure 9.16 sketches the approach to obtain a fully integrated model covering all relevant aspects of the process at hand. The approach consists of five steps. For each step we provide pointers to chapters and sections in this book

- *Step 1: obtain an event log*. Chapter 5 showed how to extract event data from a variety of systems. As explained using Fig. 5.1, this is an iterative process. The dotted chart described in Sect. 9.2 helps to explore the event log and guide the filtering process.
- *Step 2: create or discover a process model*. Chapters 6 and 7 focus on techniques for process discovery. Techniques such as heuristic mining and genetic mining can be used to obtain a process model. However, also existing hand-made models can be used.

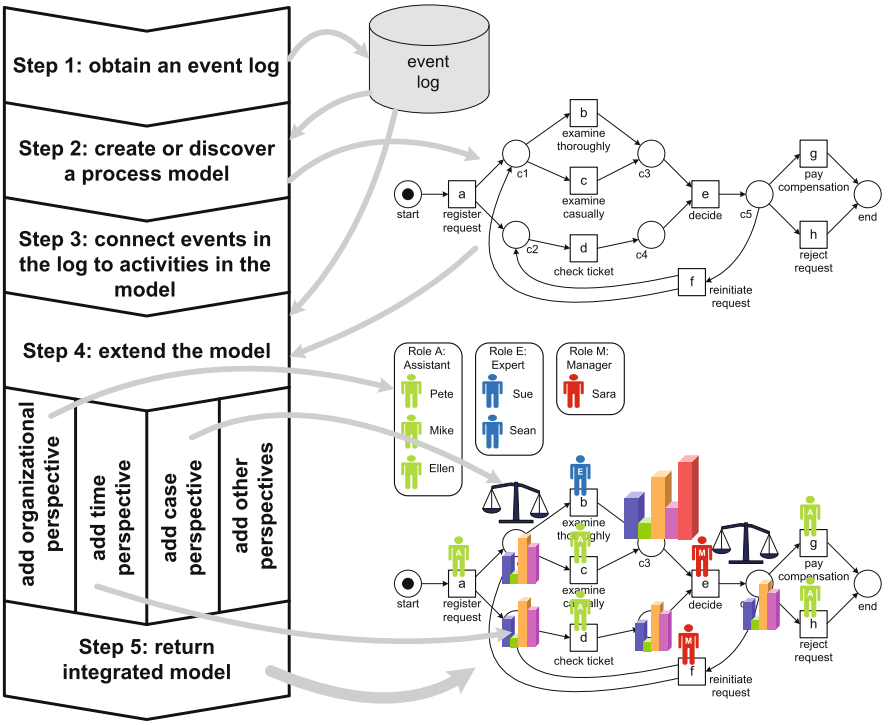

Fig. 9.16 Approach to come to a fully integrated model covering the organizational, time, and case perspectives

- *Step 3: connect events in the log to activities in the model*. As discussed in Sect. 8.5.3, this step is essential for projecting information onto models and to add perspectives. Using the replay technique described in Sect. 8.2, events in the log and activities in the model get connected.
- *Step 4: extend the model*. This is the topic of the current chapter.
  - *Step 4a: add the organizational perspective*. As shown in Sect. 9.3, it is possible to analyze the social network and subsequently identify organizational entities that connect activities to groups of resources.
  - *Step 4b: add the time perspective*. Timestamps and frequencies can be used to learn probability distributions that adequately describe waiting and service times and routing probabilities. Section 9.4 demonstrates that the replay techniques used for conformance checking can be modified to add the time perspective to process models.
  - *Step 4c: add the case perspective*. Section 9.5 showed how to use attributes in the log for decision mining. This shows which data is relevant and should be included in the model.
  - *Step 4d: add other perspectives*. Depending on the information in the log other perspectives may be added to model. For example, information on risks and

costs can be added to the model. Existing risk analysis techniques and costing approaches such as Activity Based Costing (ABC) and Resource Consumption Accounting (RCA) can be used to extend the model [31].

- *Step 5: return the integrated model*.

In Chaps. 13 and 14, we provide an overal life-cycle describing a process mining project (L^* life-cycle model). This more elaborate life-cycle incorporates Fig. 9.16.

The integrated model resulting from the steps in Fig. 9.16, can be used for various purposes. First of all, it provides a *holistic view* on the process. This provides new insights and may generate various ideas for process improvement. Moreover, the integrated model can be used as input for other tools and approaches. For instance, it can be used as a starting point for configuring a WFM or BPM system. During the configuration of such a system for a specific process, one needs to provide a model for the control-flow and the other perspectives. The integrated model can also be used to generate a simulation model covering all perspectives. For example, in [124] it is shown that the techniques described in this chapter can be used to generate a simulation model in *CPN Tools*. CPN Tools is a powerful simulation environment based on *colored Petri nets* [82, 149] (see www.cpntools.org).

The resulting simulation model closely follows reality as it is based on event logs rather than human modeling. The colored Petri net models control-flow, data flow, decisions, resources, allocation rules, service times, routing probabilities, arrival processes, etc., thus capturing all aspects relevant for simulation. The integrated simulation model can be used for "what if" analysis to explore different redesigns and control strategies.

Short-term simulation

As stressed earlier, it is essential that events in the log are connected to model elements. This allows for the projection of dynamic information onto models: the event log "breathes life" into otherwise static process models. Moreover, the merging of the various perspectives into a single model depends on this. Establishing a good connection between an event log and model may be difficult and require several iterations. However, when using a BPM system, this connection already exists; BPM systems are driven by explicit workflow models and provide excellent event logs. Moreover, internally such systems also have an explicit representation of the state of each running case. This enables a new type of simulation called *short-term simulation* [125, 139]. The key idea is to start all simulation runs from the current state and focus the analysis of the transient behavior. This way a *"fast forward button"* into the future is provided.

Figure 9.16 sketches how a simulation model could be obtained that closely matches reality. The current state obtained from the BPM system, i.e., the markings of all cases and related data elements, can be loaded into the simulation as a realistic initial state.

To understand the importance of short-term simulation, we elaborate on the difference between *transient analysis* and *steady-state analysis*. The key idea of simulation is to execute a model repeatedly. The reason for doing the experiments repeatedly, is to not come up with just a single value (e.g., "the average response time is 10.36 minutes") but to provide confidence intervals (e.g., "the average response time is with 90 percent certainty between 10 and 11 minutes"). For transient analysis the focus is on the initial part of future behavior, i.e., starting from the initial state the "near future" is explored. For transient analysis the initial state is very important. If the simulation starts in a state with long queues of work, then in the near future flow times will be long and it may take some time to get rid of the backlog. For steady-state analysis the initial state is irrelevant. Typically, the simulation is started "empty" (i.e., without any cases in progress) and only when the system is filled with cases the measurements start.

Steady-state analysis is most relevant for answering strategic and tactical questions. Transient analysis is most relevant for operational decision making. Lion's share of contemporary simulation support aims at steady-state analysis and, hence, is limited to strategic and tactical decision making. Short-term simulation focuses on operational decision making; starting from the current state—loaded from the BPM system—the "near future" is explored repeatedly [139]. This shows what will happen if no corrective actions are taken. Moreover, "what if" analysis can be used to explore the effects of different actions (e.g., adding resources and reconfiguring the process).

In [125] it is shown how this approach can be realized using the BPM system *YAWL*, the process mining tool *ProM*, and the simulation tool *CPN Tools*. This illustrates the potentially spectacular synergetic effects that can be achieved by combining workflow automation, process mining, and simulation.