Process Mining and Visual Analytics: Breathing Life into Business Process Models

Wil M.P. van der Aalst[1], Massimiliano de Leoni[1], and Arthur H.M. ter Hofstede[1,2]

[1] Eindhoven University of Technology, Eindhoven, The Netherlands
{w.m.p.v.d.aalst,m.d.leoni}@tue.nl
[2] Queensland University of Technology, Brisbane, Australia
a.terhofstede@qut.edu.au

Abstract. Process mining and visual analytics are two disciplines that emerged over the last decade. The goal of process mining is to use event data to extract process-related information, e.g., to automatically discover a process model by observing events recorded by some information system or to check the conformance of a process model with actual process executions. The spectacular growth of event data provides unprecedented opportunities and has triggered the development of a range of process mining techniques over the last decade. Despite the wonderful capabilities of existing algorithms, it has become clear that human judgment is essential in finding interesting and relevant patterns. Visual analytics combines automated analysis with interactive visualizations so as to allow decision makers to combine their flexibility, creativity, and background knowledge to come to an effective understanding of situations in the context of large data sets. This paper combines ideas from these two disciplines (i.e., process mining and visual analytics). In particular, we focus on replaying event logs on "maps" (i.e., visual representations of a process from a particular angle). If the visualization of a business process at a particular moment corresponds to "photo", then the (iterative) replay of an event log can be seen as a "movie". This way event logs can be used to "breathe life" into otherwise static process models. The insights obtained from such visualizations can be used to improve processes by removing inefficiencies and addressing non-compliance.

Key Words: process mining, visual analytics, business intelligence, business process management

1 Introduction

Process mining provides a new means to improve processes in a variety of application domains. There are two main drivers for this new technology. On the one hand, more and more events are being recorded thus providing detailed information about the history of processes. Despite the omnipresence of event data, most organizations diagnose problems based on fiction rather than facts. On the other hand, vendors of Business Process Management (BPM) and Business Intelligence (BI) software have been promising miracles. Although BPM and BI technologies received lots of attention, they did not live up to the expectations raised by academics, consultants, and software vendors. Process mining is able to effectively overcome these limitations by bridging the gap between process modeling and data mining.

There are basically three types of process mining. The first type of process mining is *discovery*. A discovery technique takes an event log and produces a model without using any a priori information. The second type of process mining is *conformance*. Here, an existing process model is compared with an event log of the same process. Conformance checking can be used to check if reality, as recorded in the log, conforms to the model and vice versa [1, 2]. The third type of process mining is *enhancement*. Here, the idea is to extend or improve an existing process model using information about the actual process recorded in some event log [1]. Whereas conformance checking measures the alignment between model and reality, this third type of process mining aims at changing or extending the a priori model. One type of enhancement is *repair*, i.e., modifying the model to better reflect reality. Another type of enhancement is *extension*, i.e., adding a new perspective to the process model by cross-correlating it with the log. An example is the extension of a process model with performance data. For instance, by using timestamps in the event log it is possible to highlight bottlenecks.

Over the last decade event data has become readily available and process mining techniques have matured. Moreover, process mining algorithms have been implemented in various academic and commercial systems. Today, there is an active group of researchers working on process mining and it has become one of the “hot topics” in BPM research. Moreover, there is a huge interest from industry in process mining. More and more software vendors started adding process mining functionality to their tools. The open-source process mining tool ProM (cf. `processmining.org`) is widely used all over the globe and provides an easy starting point for practitioners, students, and academics.

The amount of data recorded in various domains has been growing exponentially, thereby following Moore’s law. This offers opportunities for algorithmic techniques, but also creates new challenges. One of the challenges is to combine purely automatic analysis and visualization methods. While automatic algorithms for process mining and analysis are certainly needed to filter out irrelevant data and to produce preliminary results, visual inspection, domain knowledge, human judgment and creativity are needed for proper interpretation of the results. *Visual analytics* provides an answer to these problems by proposing a tight integration between automatic techniques and visualization. The term *visual analytics* was coined by Jim Thomas to mean “the science of analytical reasoning facilitated by visual interactive interfaces” [3]. Over time, the scope of visual analytics broadened. Now the term refers to a multidisciplinary field that combines elements from human-computer interaction, geo-spatial and temporal data processing, data analysis and statistics [4, 5]. A more recent definition is “Visual analytics combines automated analysis techniques with interactive visualizations for an effective understanding, reasoning and decision making on the basis of very large and complex data sets” [5].

Figure 1 positions visual analytics. Traditionally, visual analytics combined visualization with data mining techniques [5]. However, given the maturity of process mining techniques and the interest in Business Process Management (BPM) and Business Intelligence (BI), it makes sense to develop visual analytics based on process mining. Visual

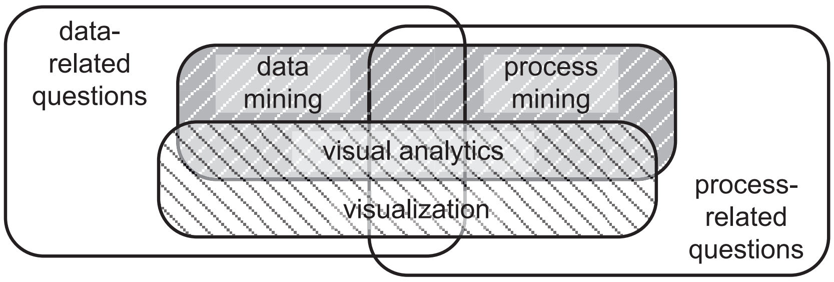


Fig. 1. Visual analytics combines analytical techniques (e.g., data mining) and visualization. Whereas data mining aims to answer data-related questions, process mining aims to answer process-related questions.

analytics have established a synergy between visualization and data mining. Now it is time to realize similar synergetic effects between visualization and process mining. Examples such as the *Dotted chart* and the *Fuzzy miner* illustrate the importance of the relation between process mining and visualization. The *Dotted chart* provides a helicopter view of the process and is used as the starting point for any process mining project [1]. In a Dotted chart, each event is depicted as a colored dot in a two dimensional plane. This is used to filter the event log and often immediately provides interesting insights. Another example is the so-called *Fuzzy miner* [6] which interactively creates a model using the metaphor of a map. In this approach process models are used as if they are geographic maps (e.g., road maps or hiking maps). Depending on the map, insignificant roads and cities can be removed and streets and suburbs can be amalgamated into bigger structures.

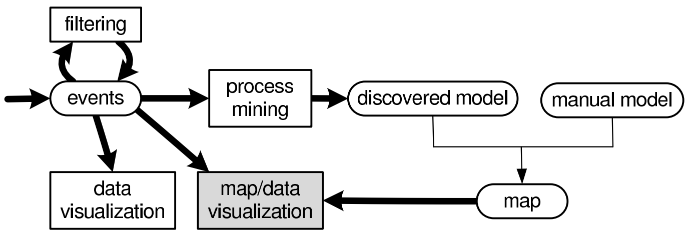


Fig. 2. The focus of this paper is on the visualization of event-related data projected on models obtained through process mining.

Figure 2 shows a high-level characterization of visual analytics based on process mining. Assume that one has an event log, i.e., large collections of events ordered in time. The events in this log, i.e., the input data, can be visualized directly (e.g., using Dotted charts). Event logs can also be converted into models (e.g., process models, organizational models, decision models, or predictive models), which can be visualized in a static manner. For instance, the model can show the discovered process, e.g. represented by an UML activity diagram or BPMN diagram. However, the event log and model can also be combined and visualized. For example, the event log can be replayed on a discovered process model. In general, the concept of model can be interpreted in a broader sense: in principle, any flat image can be considered as a model. In fact, models can take different forms, such as a geographic map, a Gantt chart, an organizational diagram, a social network, a depiction of various deadlines and several more.

In general a model, discovered or designed by hand, can be interpreted as a *map* which focuses on showing a specific view on the process (the process control flow, the organizational structure, the geo-spatial or temporal characteristics of the process and its composing tasks, etc.). After each event, the process is in a particular state, which can be projected onto such defined maps. For instance, if the map shows the chart of the different offices, the state is projected as a set of dots, one per task; each dot is located onto the office in the map where the respective task needs to be executed. Vice versa, if the map is an organization chart, every dot represents one task and is placed on the role that participants need to play in order to execute it.

The projection onto maps is a kind of *photograph* of the process, seen from a certain viewpoint (e.g., the organization or the geographic perspective). Since such a photograph is available after each event, for each map considered, it is possible to create a *"movie"* by showing the photographs in succession. Movies can be played at different speeds and users are allowed to switch from one "movie" to another at any time. Through such "movies", process analysts can evaluate the performance of past process instances from different views. The amount of information shown in the "movie" is customizable (e.g., a process analyst may be interested to visualize all the work performed by all participants or simply that of specific individuals). Indeed, by interacting with the user interface, the analysts can "drill down" into the piece of information in which they are interested (e.g., they may want to get a deeper insight into the work performed by a specific participant) or "roll up" to have an aggregated view (e.g., the work performed by a larger group of participants). When rolling up, some information is automatically filtered out to simplify the interpretation of the "movie". Using this movie metaphor, process analysts can see behaviors and issues that would remain undetected using purely algorithmic techniques. For instance, let us consider the movie related to an organization chart: if the dots that correspond to work that needs to be done by people having a specific role seem to be cluttering the diagram during movie replay, then probably process performance can be improved by assigning that role to a larger number of participants.

Although Figure 2 may suggest the application of visual analytics to process mining to be a waterfall approach, it is actually iterative. The visualization of models and the projection of the current and past states onto maps can lead to the desired information. Nonetheless, it is more probable that the preliminary results need to be refined and, hence, some parameters need further tuning. For instance, it may be the case that the event filtering, the process mining techniques or, simply, the visualization need to be adjusted further. Therefore, several iterations are, in general, needed, alternating between fully-automatic techniques and visualization approaches.

The remainder of this paper is organized as follows. Section 2 provides an overview of process mining using a concrete example. Section 3 shows the analogy between a certain type of process representation and cartographic maps, pointing out the importance of the visualization of the different business process views. This section illustrates initial ideas using the fuzzy mining technique and then shows how the discovered process models can be used as maps for which movies are constructed. Section 4 describes how to use information in the log to a posteriori build the photographs, i.e. the projections of the different states onto maps, and to play the photographs in succession for obtaining the movies. By playing these movies, process analysts can evaluate past process executions from different perspectives encoded in corresponding maps so as to find issues and bottlenecks. Section 5 briefly describes existing tool support for process mining, whereas related work is discussed in Section 6. Finally, Section 7 concludes this paper by outlining future research directions.

2 Process Mining

To introduce the basic idea of process mining, we use an example taken from [1]. Table 1 shows just a fragment of a possible *event log*. Each line captures one *event*. Events are already grouped per *case*. Each case corresponds to a *process instance*, i.e., one run of the process from beginning to end. In Table 1, each case corresponds to the handling of a request for compensation. Case 1 has five associated events. The first event of Case 1 is the execution of activity *register request* by Pete on December 30th 2010. Table 1 also shows a unique id for this event: 35654423. This is merely used for the identification of the event, e.g., to distinguish it from event 35654483 that also corresponds to the execution of activity *register request* (first event of second case). Table 1 shows a date and a timestamp for each event. In some events logs this information is more coarse-grained and only a date or partial ordering of events is given. In other logs there may be more elaborate timing information also showing when the activity was started, when it was completed, and sometimes even when it was offered to the resource. The times shown in Table 1 should be interpreted as completion times. In this particular event log, activities are considered to be atomic and the table does not reveal the duration of activities. In the table, each event is associated to a resource. In some event logs this information may be missing. In other logs more detailed information about resources may be stored, e.g., the role a resource has or elaborate authorization data. The table also shows the costs associated with events. This is an example of a data attribute. There may be many other data attributes. For example, in this particular example it would be interesting to record the outcome of the different types of examinations and checks. Another data element that could be useful for analysis is the amount of compensation requested. This could be an attribute of the whole case or stored as an attribute of the *register request* event.

Depending on the process mining technique employed and the questions at hand, only part of the information in the event log is used. The minimal requirements for process mining are that any event can be related to both a case and an activity and that events within a case are ordered. Hence, the “case id” and “activity” columns in Table 1 represent the bare minimum for process mining. By projecting the information in these two columns we obtain the more compact representation shown in Table 2. In this table, each case is represented by a sequence of activities also referred to as a *trace*.

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	...
6	35654871	06-01-2011:15.02	register request	Mike	50	...
	35654873	06-01-2011:16.06	examine casually	Ellen	400	...
	35654874	07-01-2011:16.22	check ticket	Mike	100	...
	35654875	07-01-2011:16.52	decide	Sara	200	...
	35654877	16-01-2011:11.47	pay compensation	Mike	200	...
5	35654711	06-01-2011:09.02	register request	Ellen	50	...
	35654712	07-01-2011:10.16	examine casually	Mike	400	...
	35654714	08-01-2011:11.22	check ticket	Pete	100	...
	35654715	10-01-2011:13.28	decide	Sara	200	...
	35654716	11-01-2011:16.18	reinitiate request	Sara	200	...
	35654718	14-01-2011:14.33	check ticket	Ellen	100	...
	35654719	16-01-2011:15.50	examine casually	Mike	400	...
	35654720	19-01-2011:11.18	decide	Sara	200	...
	35654721	20-01-2011:12.48	reinitiate request	Sara	200	...
	35654722	21-01-2011:09.06	examine thoroughly	Sue	400	...
	35654724	21-01-2011:11.34	check ticket	Pete	100	...
	35654725	23-01-2011:13.12	decide	Sara	200	...
	35654726	24-01-2011:14.56	reject request	Mike	200	...
...	...	...	...	...	...	...

Table 1. A fragment of some event log: each line corresponds to an event.

case id	trace
1	$\langle a, b, d, e, h\rangle$
2	$\langle a, d, c, e, g\rangle$
3	$\langle a, c, d, e, f, b, d, e, g\rangle$
4	$\langle a, d, b, e, h\rangle$
5	$\langle a, c, d, e, g\rangle$
6	$\langle a, c, d, e, f, d, c, e, f, b, d, e, h\rangle$
...	...

Table 2. A more compact representation of the log shown in Table 1: $a =$ *register request*, $b =$ *examine thoroughly*, $c =$ *examine casually*, $d =$ *check ticket*, $e =$ *decide*, $f =$ *reinitiate request*, $g =$ *pay compensation*, and $h =$ *reject request*.

For clarity the activity names have been transformed into single-letter labels, e.g., a denotes activity *register request*.

Process mining algorithms for process discovery can transform the information shown in Table 2 into process models. For instance, the process model in Figure 3 can be the result of applying some basic process discovery technique [1]. For example, the α-algorithm [7, 1] is able to discover a Petri net based on a sequence of events. Heuristic mining [8, 1] is an enhanced approach that can deal with noisy logs, in the sense that it is able to distinguish between low frequency behaviors, i.e. the noise, and high frequency ones. Indeed, many algorithms consider all behaviors in the same way and, hence, the mined model allows for many behaviors, of which some are occurring many times and others a few times in the event log. Some of these infrequent behaviors recorded in the log can be the result of erroneous process performances or of exceptional conditions which required the activation of special procedures. The presence of these infrequent behaviors can compromise model readability. When showing the mainstream behavior the diagram should not be cluttered by infrequent/exceptional behavior. The genetic process mining approach presented in [9, 1] proposes algorithms that incrementally try to find the most suitable process model. In the first step, possible suitable process models are generated and their fitness is computed. The models that fit best are kept for the next step; the least fitting models are removed. Some of the best models are merged and mutated through specific operators and the models resulting from crossover and mutation are considered for the next step. The algorithm progresses through potentially thousands of iterations and stops when no further improvements can be found. Then the best fitting process model is presented as the final result. The generic mining is robust against noise (just like heuristic mining) and can be combined with other approaches. Genetic mining can be very time consuming. However, the approach can easily be parallelized using a grid infrastructure.

Let us consider again the model in Figure 3. Given an event log $L = \{\langle a, b, d, e, h\rangle, \langle a, d, c, e, g\rangle, \langle a, c, d, e, f, b, d, e, g\rangle, \langle a, d, b, e, h\rangle, \langle a, c, d, e, g\rangle, \langle a, c, d, e, f, d, c, e, f, b, d, e, h\rangle\}$ (i.e., Table 2), most process discovery algorithms will construct this model (although the representation may be different). It is easy to check that all six traces in L *fit* (i.e., conform to) the model since they are are possible in the model. Typically, the trace conformance is based on the *principle of replay*, i.e., the event log is replayed on the process model. The events in the traces are mapped to a task in the process model; when a trace is replayed, events are sorted by their timestamp. Tasks that belong to logged events in the trace are executed in the order recorded. If the trace does not fit completely, then some tasks need to be executed according to the log, while this is not possible according to the model. For instance, a task is not allowed to occur since some tasks that precede in the model have not occurred in the event log. There are different ways to measure errors in the course of replaying; see [1, 2] for metrics to measure the degree of fitness of a trace with respect to a process model.

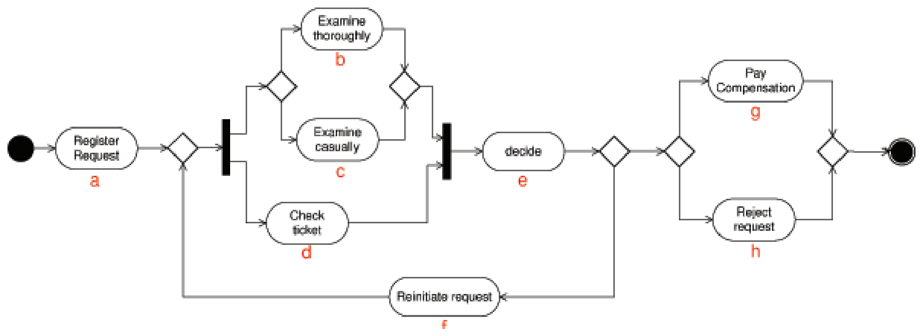


Fig. 3. A model that can be obtained by some basic process discovery technique. The discovered model is based on the set of traces shown in Table 2.

The process model shown in Figure 3 also allows for traces not present in Table 2. For example, the traces $\langle a, d, c, e, f, b, d, e, g \rangle$ and $\langle a, c, d, e, f, c, d, e, f, c, d, e, f, c, d, e, f, b, d, e, g \rangle$ are also possible. This is a desired phenomenon as the goal is *not* to represent just the *particular set of example traces* in the event log. Process mining algorithms need to generalize the behavior contained in the log to show the most likely underlying model that is not invalidated by the next set of observations. One of the challenges of process mining is to balance between "overfitting" (the model is too specific and only allows for the "accidental behavior" observed) and "underfitting" (the model is too general and allows for behavior unrelated to the behavior observed) [1].

As explained in Section 1, process mining is not limited to process discovery. Event logs can be used to check conformance and enhance existing models. Moreover, different perspectives may be taken into account. To illustrate this, let us first consider the event log shown in Table 3. The first six cases are as before. It is easy to see that Case 7 with trace $\langle a, b, e, g \rangle$ is not possible according to the model in Figure 3. The model requires the execution of d before e, but d did not occur. This means that the ticket was not checked at all before making a decision and paying compensation. Conformance checking techniques aim at discovering such discrepancies [2]. When checking the conformance of the remainder of the event log it can be noted that cases 8 and 10 do not conform either. Case 9 conforms although it is not identical to one of the earlier

case id	trace
1	$\langle a, b, d, e, h \rangle$
2	$\langle a, d, c, e, g \rangle$
3	$\langle a, c, d, e, f, b, d, e, g \rangle$
4	$\langle a, d, b, e, h \rangle$
5	$\langle a, c, d, e, g \rangle$
6	$\langle a, c, d, e, f, d, c, e, f, b, d, e, h \rangle$
7	$\langle \mathbf{a}, \mathbf{b}, \mathbf{e}, \mathbf{g} \rangle$
8	$\langle \mathbf{a}, \mathbf{b}, \mathbf{d}, \mathbf{e} \rangle$
9	$\langle a, d, c, e, f, d, c, e, f, b, d, e, h \rangle$
10	$\langle \mathbf{a}, \mathbf{c}, \mathbf{d}, \mathbf{e}, \mathbf{f}, \mathbf{b}, \mathbf{d}, \mathbf{g} \rangle$

Table 3. Another event log: cases 7, 8, and 10 are not possible according to Figure 3.

traces. Trace $\langle a, b, d, e \rangle$ (i.e. Case 8) has the problem that no concluding action was taken (rejection or payment). Trace $\langle a, c, d, e, f, b, d, g \rangle$ (Case 10) has the problem that the airline paid compensation without making a final decision. Note that conformance can be viewed from two angles: (a) the model does not capture the real behavior ("the model is wrong") and (b) reality deviates from the desired model ("the event log is wrong"). The first viewpoint is taken when the model is supposed to be *descriptive*, i.e., capture or predict reality. The second viewpoint is taken when the model is *normative*, i.e., used to influence or control reality.

The original event log shown in Table 1 also contains information about resources, timestamps and costs. Such information can be used to discover other perspectives, check the conformance of models that are not solely control-flow models, and to extend models with additional information. For example, one could derive a social network based on the interaction patterns between individuals. The social network can be based on the "handover of work" metric, i.e., the more often individual x performed an activity that is causally followed by an activity performed by individual y, the stronger the relation between x and y is [10].

Figure 4 illustrates in what ways a control-flow oriented model can be extended with the other perspectives (data perspective, resource/organizational perspective, time perspective). Analysis of the event log shown in Table 1 may reveal that Sara is the only one performing the activities *decide* and *reinitiate* request. This suggests that there is a "manager role" and that Sara is the only one having this role. Activity *examine thoroughly* is performed only by Sue and Sean. This suggests some "expert role" associated to this activity. The remaining activities are performed by Pete, Mike and Ellen. This suggests some "assistant role" as shown in Figure 4. Techniques for organizational process mining [10] will discover such organizational structures and relate activities to resources through roles. By exploiting resource information in the log, the organizational perspective can be added to the process model. Similarly, information on timestamps and frequencies can be used to add performance related information to the model. Figure 4 sketches that it is possible to measure the time that passes between an examination (activities b or c) and the actual decision (activity e). If this time is remarkably long, process mining can be used to identify the problem and discover possible causes.

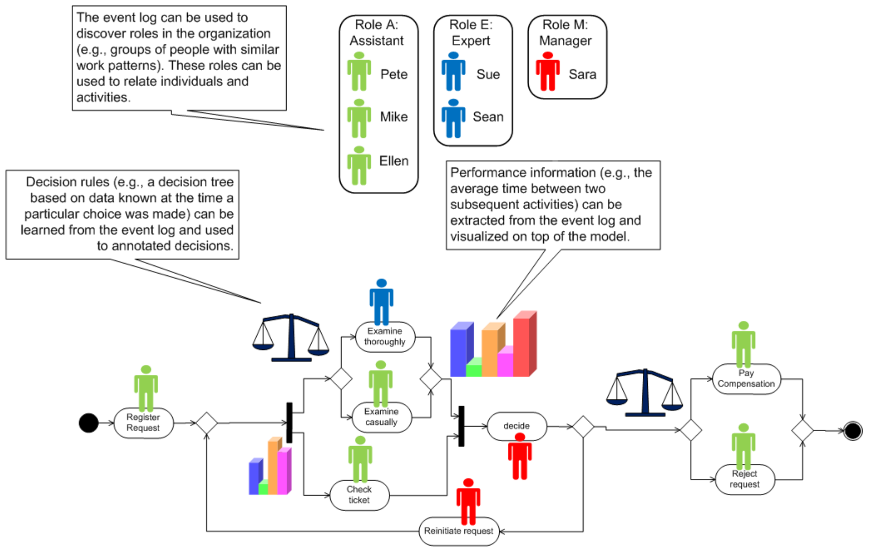

The event log can be used to discover roles in the organization (e.g., groups of people with similar work patterns). These roles can be used to relate individuals and activities.
Role A: Assistant
Pete
Mike
Ellen
Role E: Expert
Sue
Sean
Role M: Manager
Sara
Decision rules (e.g., a decision tree based on data known at the time a particular choice was made) can be learned from the event log and used to annotated decisions.
Performance information (e.g., the average time between two subsequent activities) can be extracted from the event log and visualized on top of the model.
Register Request
Examine thoroughly
Examine casually
Check ticket
decide
Reinitiate request
Pay Compensation
Reject request

Fig. 4. The process model extended with additional perspectives: the organizational perspective ("What are the organizational roles and which resources are performing particular activities?"), the case perspective ("Which characteristics of a case influence a particular decision?"), and the time perspective ("Where are the bottlenecks in my process?").

If the event log contains case-related information, this can be used to further analyze the decision points in the process. For instance, through decision point analysis it may be learned that requests for compensation of more than € 800 tend to be rejected.

Using process mining, the different perspectives can be cross-correlated to find surprising insights [1]. Examples of such findings could be: "requests examined by Sean tend to be rejected more frequently", "requests for which the ticket is checked after examination tend to take much longer", "requests of less than € 500 tend to be completed without any additional iterations". Moreover, these perspectives can also be linked to conformance questions. For example, it may be shown that Pete is involved in relatively many incorrectly handled requests.

3 Process Visualization: The Map Metaphor

Models—discovered by process mining techniques or made by hand—can be seen as the "maps" describing the operational processes of organizations. Cartography evolved over many centuries. Today, there are standard techniques to create maps thereby addressing problems such as clearly representing desired traits, eliminating irrelevant details, reducing complexity, and improving understandability. Generally, maps are of high quality and can easily be understood by humans. Moreover, modern car navigation systems (e.g., TomTom, Garmin, and Navigon), Google Maps, and mashups using geo-tagging show innovative ways of using maps. All of this is stark contrast with the state-of-the-art in business process modeling. Accurate business process maps are typically missing. Process models tend to be outdated and not aligned with reality. Moreover, unlike geographic maps, process models are typically not well understood by end users.

Process mining techniques—specifically process discovery and conformance checking—can be used to improve the accuracy of maps. However, more accurate models are not necessarily better models. Therefore, it is interesting to see what one can learn from maps.

Figure 5 shows a map. The map *abstracts* from less significant roads and cities. Roads that are less important are not shown. A cut-off criterion could be based on the average number of cars using the road per day. Similarly, the number of citizens could be used as a cut-off criterion for cities. For example, in Figure 5 cities of less than 50,000 inhabitants are abstracted from. Maps also *aggregate* local roads and local districts (neighborhoods, suburbs, centers, etc.) into bigger entities. Figure 5, for instance, shows Eindhoven as a single dot while it consists of many roads, various districts (Strijp, Gestel, Woensel, Gestel, etc.), and neighboring cities (e.g., Veldhoven). People interested in Eindhoven can look at a city map to see more details.

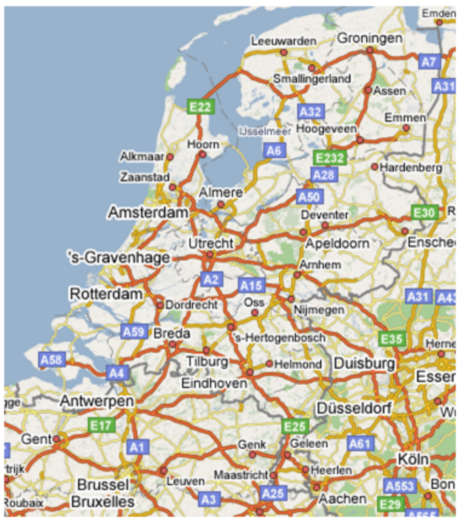


Fig. 5. Road map of The Netherlands. The map abstracts from smaller cities and less significant roads; only the bigger cities, highways, and other important roads are shown. Moreover, cities aggregate local roads and local districts.

Models that describe views on processes need to abstract from less significant things. Activities can be removed if they are less frequent, e.g., activities that occur in less than 20% of completed cases are abstracted from. Also time and costs can be taken into account, e.g., activities that account for less than 8% of the total service time are removed unless the associated costs are more than € 50,000.

There may be different geographic maps of the same area using different scales. Moreover, using electronic maps it is possible to seamlessly zoom in and out. Note that, while zooming out, insignificant things are either left out or dynamically clustered into aggregate shapes (e.g., streets and suburbs amalgamate into cities). Navigation systems and applications such as Google Maps provide such a seamless zoom. Traditionally, process models are static, e.g., it is impossible to seamlessly zoom in to see part of the process in more detail. To deal with larger processes, typically a static hierarchical decomposition is used. Such a decomposition does not allow for a seamless abstraction. Moreover, the hierarchy forces people to see less significant lower-level connections at the highest level [1].

Cartographers not only eliminate irrelevant details, but also use colors to highlight important features. For instance, the map shown in Figure 5 emphasizes the importance of highways using the color red. Moreover, graphical elements have a particular size to indicate their significance, e.g., the sizes of lines and dots may vary. For instance, in Figure 5 the size of a city name is proportional to the number of citizens, e.g., Zaanstad is clearly smaller than Amsterdam. Geographical maps also have a clear interpretation of the x-axis and y-axis, i.e., the layout of a map is not arbitrary as the coordinates of elements have a meaning.

Ideas from cartography can easily be incorporated in the construction of maps. Some examples:

- The *size of an activity* can reflect its frequency or some other property indicating its significance (e.g., costs or resource use) or the time required for its performance.
- The *color of an activity* can reflect the state (e.g., whether it is started, assigned or concluded).
- The *width of an arc* can reflect the importance of the corresponding causal dependency among activities.
- The *coloring of arcs* can be used to highlight bottlenecks.
- The *positioning of activities* can have a well-defined meaning. Similar to swimlanes the y-axis could reflect the role associated to an activity. Similar to a Gantt chart, the x-axis could reflect some temporal aspect.

Section 4 describes ongoing research that employs some of these ideas with the aim of developing a tool to allow for analytical reasoning of the past execution of business processes. The process of reasoning will be supported by interactive graphical interfaces that provide several visual representations of the process execution and data involved.

In the remainder of this section we illustrate some of the ideas using *fuzzy mining* [6]. First, we apply ideas obtained from cartography to build process-model maps (Section 3.1). Then, we show the use of animation as a tool for gaining a better understanding of the actual behavior found in event logs (Section 3.2).

3.1 Fuzzy Mining

Various real-life applications of the different traditional process mining techniques have shown that the discovered models often look like “spaghetti”, showing all details without highlighting what is important. Indeed, as previously said in Section 2, logs may usually include less-relevant or infrequent behavior. In [6] the authors propose *fuzzy mining* as a new process mining approach to avoid spaghetti-like models that are incomprehensible. In the same way as roadmaps provide suitable abstractions of reality, process models should provide meaningful abstractions of operational processes.

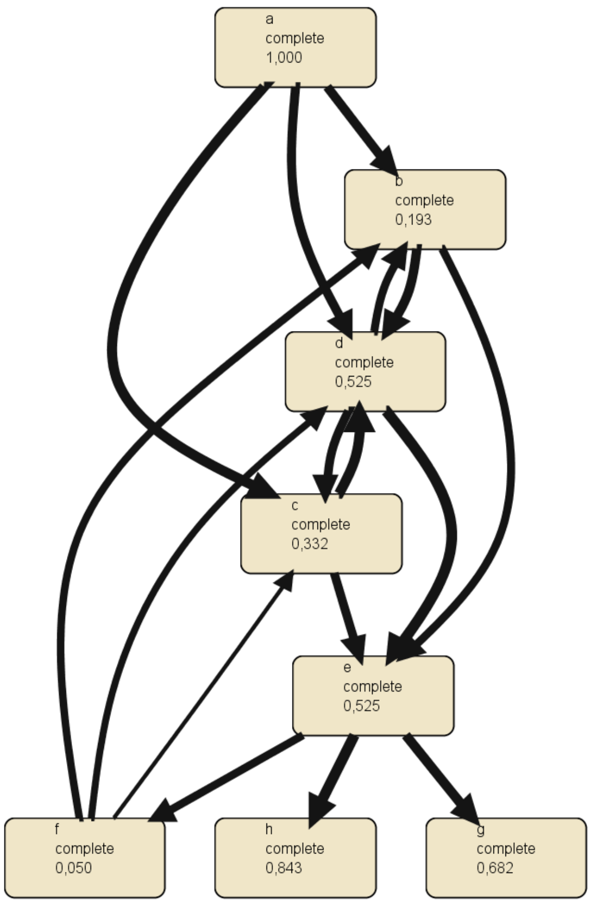


Fig. 6. Fuzzy model based on the event log shown in Table 1. The log contains 7539 events corresponding to 1391 process instances. The most detailed view is shown, i.e., all activities are included in the model.

Figure 6 shows a fuzzy model corresponding to the event log that was used to construct the process model of Figure 3. This model shows all activities and all causal dependencies. It can be seen as the most detailed map of the process.

By moving a slider in the Fuzzy miner of ProM, the less frequent or less important activities can be removed. The simplest model would only show the most frequent activity and abstract from all other activities. Less frequent activities can be removed

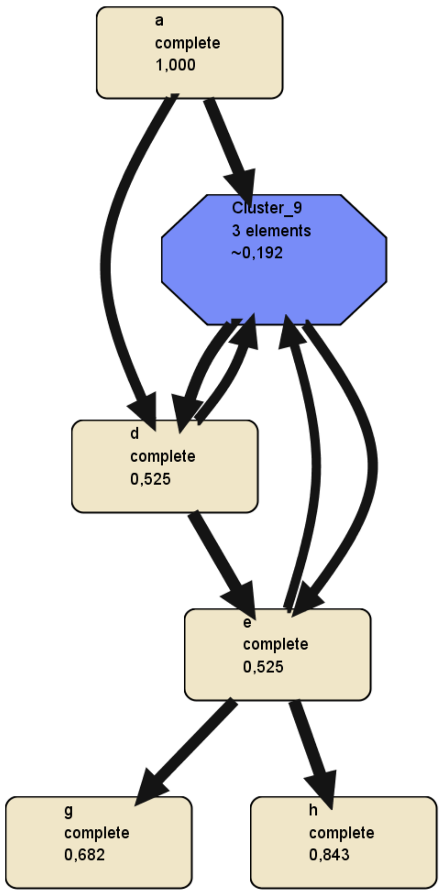


Fig. 7. Fuzzy model where only the five most frequent activities are shown at the highest level of abstraction. The cluster node contains three less frequent, but correlated, activities.

or hidden in so-called cluster nodes. Figure 7 shows a fuzzy model based on the same event log used to construct Figure 6. However, the three least frequent activities are hidden in a cluster node. One can think of such a cluster node as a subprocess. The activities in a cluster node are related, but not important enough to be shown at the top level. A cluster node can be seen as a “city map”.

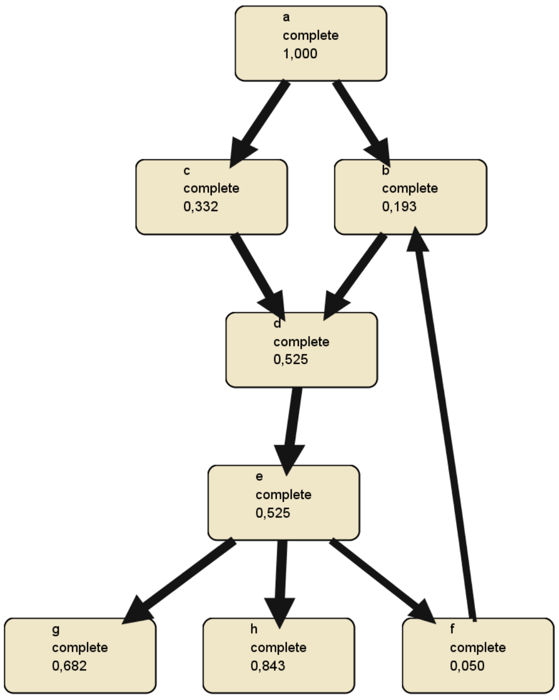


Fig. 8. Fuzzy model where all activities are included at the highest level of abstraction. However, the less frequent paths in the model are removed. Although b, c, or d can directly precede e, in most cases d directly precedes e. Although f may be followed by b, c and/or d, it is followed most frequently by b.

Less frequent activities can be removed or hidden at a lower level. Similarly, one can remove less frequent connections in a process model. Figure 8 shows a fuzzy model where only the most important connections are shown. Note that Figure 6 is based on the same event log but has more arcs than the fuzzy model in Figure 8. Using the metaphor of a roadmap, one could say that Figure 8 only shows the highways whereas Figure 6 also shows less important roads.

The Fuzzy miner of ProM has many parameters to seamlessly simplify models [6]. For our simple example there is no need to simplify the model. However, real-life processes tend to be spaghetti-like involving dozens of activities and hundreds of connections. Such models cannot be understood by end-users. Therefore, simplification is essential, just like a map not showing all details.

3.2 Fuzzy Animation

The actual process behavior, as found in the event logs, can be projected onto fuzzy models. The result is a movie that can be played in order to come up with a better understanding of what has occurred in reality [6]. In this way, while watching the movie, a process analyst becomes aware which parts of the model are important and where problems occur.

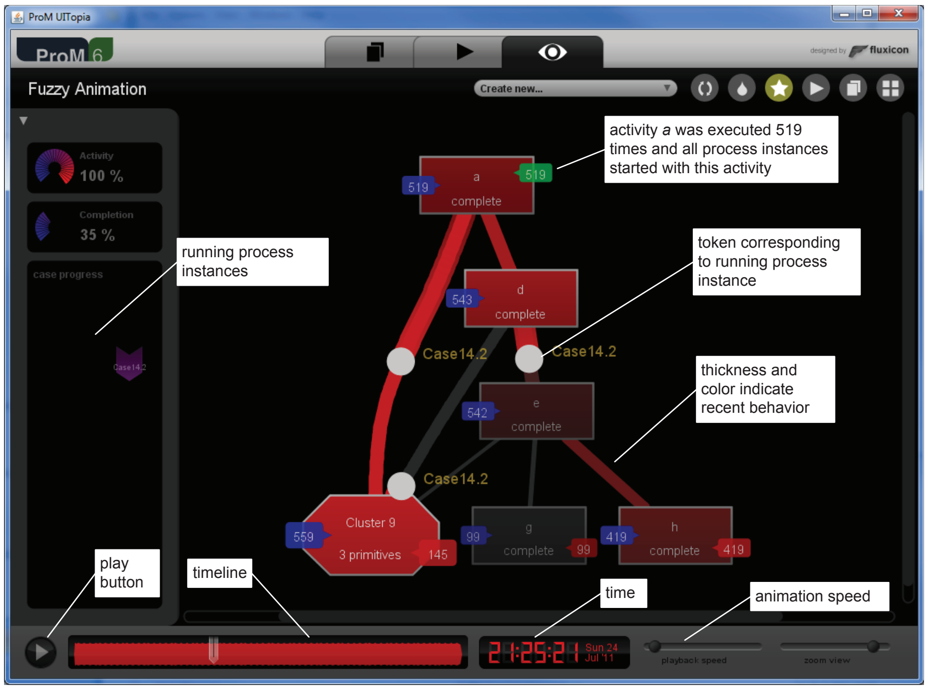


Fig. 9. Screenshot of the Fuzzy miner while animating the event log consisting of 7539 events related to 1391 process instances. Note that unlike the animations in simulation tools, the animation is based on factual data. For example, it is possible to follow concrete cases. Moreover, it is easy to see where in the process bottlenecks appear and disappear over time.

Figure 9 shows the fuzzy model animation described in [6, 11] and implemented in the ProM framework (see also Section 5). Replaying an event log can be done by simply passing control from one node to another in the fuzzy model through one of the available arcs. As the net is replayed based on information in the log, no executable semantics are needed. However, while replaying, the nature of splits and joins becomes clear. If control passes from a node A to another node B, then this is visualized through a token that moves along the arc from A to B. When passing along an arc, the token leaves a trail of glowing hot particles as it were a comet. The approach projects all log traces onto the model at once, resulting in multiple traces animated at the same time. The colors of the connections and the thickness of the arcs indicate recent activity. In this way, users can distinguish individual process instances and see the overall activity at a particular point in time. For example, a path that is being traveled frequently becomes wider while the animation progresses. Therefore, every token that travels along arcs will help to increase the thickness of the arc itself. Hence, again the map metaphor is used: on a road map highways are shown using thicker lines.

The Fuzzy miner in ProM illustrates some important observations:

- Process models can be seen as maps. Hence, we should try to use ideas from cartography to simplify and clarify processes.
- Given a process or event log there may be many different maps depending on the intended use and the aspects of interest.
- By projecting states and events on such maps, we can "breathe life" into process models. Showing an animation based on historic data reveals problems and insights that cannot be seen by looking at static graphs and charts.

Based on these observations, we propose an approach to relate visual analytics to process mining as is shown next.

4 Visual Spatio-temporal Analysis of Process Executions

This section concerns a framework that leverages the map metaphor to allow process analysts to replay history on a map to identify bottlenecks and learn more about the actual process. Specifically, the event log is replayed: after the occurrence of every event, the system state changes and, hence, the framework builds the sequence of states which the system has gone through. The framework envisages the existence of several maps; each one showing a different process view on which the system state can be projected.

After the occurrence of every event, the system state changes and, hence, its projection onto maps results in a new photograph. For each map, the different photographs can be ordered in a sequence and merged in succession; the final result is a set of movies. Each movie allows process analysts to view the process executions from a particular angle, thus highlighting some distinctive peculiarities that are less evident in the other perspectives.

4.1 The Framework Architecture

Figure 10 shows the architecture of the framework. The framework assumes that there exists an environment in which processes are executed; this environment is, in general, an information system which is not necessarily process-oriented. The only assumption is that the execution of a process instance can be broken down into the sequence of steps, i.e., its activity instances, that are carried out during process performance, and such steps are recorded and exported into an *Execution Log*.

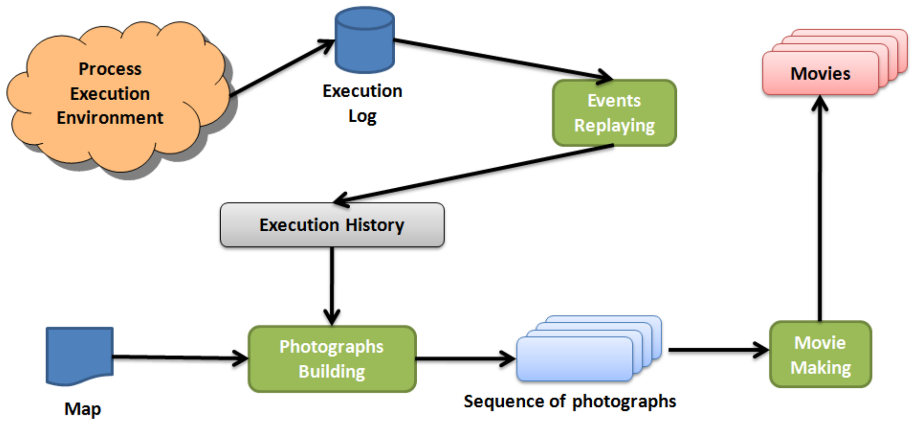


Fig. 10. Architecture of the framework for the analysis of process executions.

The events in the log can be replayed, thus allowing for rebuilding the *Execution History*. An execution history is a sequence of states which the system went through together with the timestamps marking the times when the system entered these states.

Once the execution history has been reconstructed using replay, it is possible to create a sequence of photographs. In addition to the execution history, this step requires as input the set M of maps of interest, and the functions that determine how to project single activity instances onto maps. The result is a sequence of photographs for each map $m \in M$. One photograph is built after the occurrence of every event for all maps.

Finally, each sequence of photographs is turned into a movie by playing them in succession. In general, a process execution can last days, even months. Therefore, if a photograph was built after each event, the movie would be far too long. Therefore, the step of making the movies is more than just playing photographs in succession. Photographs need to be grouped in a number of "aggregated" photographs. Each aggregated photograph summarizes the information of its constituent photographs. One can think of this as a "refresh rate": the process time is divided in periods and all photographs falling in the same time period are merged. Also note that the duration of activity instances could be very short compared to the overall flow times. In fact, in some cases activities are atomic and a naïve concatenation of steps would not show these (as they take no time).

Section 4.2 discusses the execution history, whereas Section 4.3 describes the projection of states to maps in order to obtain photographs. Finally, Section 4.4 illustrates how to obtain a movie by merging photographs and playing them in succession.

4.2 Building the Execution History

An *activity instance* refers to the execution of an activity for a specific instance. There may be multiple events that describe the life cycle of such an activity instance. The goal is to visualize how activity instances evolve over time. An activity instance is represented by a pair (a_{name}, a_{pid}) where $a_{name} \in A$ and $a_{pid} \in C$. A is the universe of activity names. C is the universe of process identifiers, i.e., a_{pid} refers to a particular process instance (sometimes referred to as a case). A process data variable is a pair (v_{name}, v_{pid}) where $v_{name} \in V$ and $v_{pid} \in C$ where V denotes the universe of variable names. Note that different process instances can have different values for some variable v_{name}.

The possible states in which an activity instance can be are the following:

- *Unscheduled.* The activity instance is not yet scheduled for execution; this is the initial state.
- *Scheduled.* The activity instance is scheduled for execution.
- *Assigned.* The activity instance is assigned to a resource for execution.
- *Executing.* The execution of the activity instance started.
- *Suspended.* The execution of the activity instance is paused.
- *Concluded.* The execution of the activity instance is concluded.

In the following, Z denotes the set of possible activity instance states: $Z = \{\mathit{Unscheduled}, \mathit{Scheduled}, \mathit{Assigned}, \mathit{Executing}, \mathit{Suspended}, \mathit{Concluded}\}$. As mentioned in Section 2, each event is recorded in the execution log together with the timestamp when it occurred:

Definition 1 (Event). *An event e is a tuple (a_{name}, cid, t, z, P) where:*

- *a_{name} is an activity name;*
- *cid is a case identifier;*
- *t is the timestamp when event e occurred;*
- *$z \in Z$ is the state to which the corresponding activity instance moves, as a result of the occurrence of event e.*
- *$P = \{v_1, \dots, v_n\}$ is a set of data properties. Every data property v_i is a pair (vn_i, val_i) where vn_i is a property name and val_i is a property value.*

The events stored in the execution logs trigger changes in the system state. A system state is defined as follows:

Definition 2 (System State). *Let Z be the activity instance states. Let A, C and V be the universe of activity names, process identifiers and variable names, respectively. U is the universe of variable values. A system state $S = (\alpha, \nu)$ consists of:*

- *a function $\alpha : (A \times C) \not\rightarrow Z$ where $\alpha(a_{name}, a_{pid}) = z$ denotes that activity instance (a_{name}, a_{pid}) is in state $z \in Z$ when the system state is S.*
- *a function $\nu : (V \times C) \not\rightarrow U$ where $\nu(v_{name}, v_{pid}) = v_{value}$ denotes that variable (v_{name}, v_{pid}) has value v_{value} in system state S.*

In the remainder, for each event $e = (a_{name}, cid, t, z, P)$, we use the following functions to refer to the elements of this tuple: $\mathit{activity}(e) = a_{name}$, $\mathit{case}(e) = cid$, $\mathit{timestamp}(e) = t$, $\mathit{state}(e) = z$ and $\mathit{properties}(e) = P$. Moreover, given a function f, we denote with $dom(f)$ the domain of function f.

Let $f(\phi_1, \dots, \phi_n)$ be an $n-$ary function and let $\Phi = \{\phi_1, \dots, \phi_m\}$ be a set of pairs $\phi_j = (\boldsymbol{\phi}^j, v^j)$ where $\boldsymbol{\phi}^j = (\phi_1^j, \dots, \phi_n^j)$ is a n-dimensional vector and v^j is an arbitrary value. We denote with $f' = f \oplus \Phi$ the function:

$$f'(\boldsymbol{\phi}) = \begin{cases} f(\boldsymbol{\phi}) & \text{if } \neg\exists v.(\boldsymbol{\phi}, v) \in \Phi \\ v & \text{if } (\boldsymbol{\phi}, v) \in \Phi \end{cases}$$

Moreover, let $\Psi = \{\boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_m\}$ be a set of n-dimensional vectors $\boldsymbol{\psi}_j = (\psi^j_1, \ldots, \psi^j_n)$. We denote with $f' = f \ominus \Psi$ the function:

$$f'(\boldsymbol{\psi}) = \begin{cases} f(\boldsymbol{\psi}) & \text{if } \boldsymbol{\psi} \notin \Psi \\ \text{undefined otherwise} \end{cases}$$

Similarly to existing algorithms for conformance checking [1], this framework is based on the principle of replay. Nevertheless, the replay used here is slightly different as will be shown next. The initial replay state is $S_0 = (\alpha_0, \nu_0)$ where $dom(\alpha_0) = \emptyset$ and $dom(\nu_0) = \emptyset$.

Definition 3 (Replaying of events). *Let $S = (\alpha, \nu)$ be the current state during the replay. Let e be the next event to replay. The replaying of event e will cause the replay state to move from state S to state $S' = (\alpha', \nu')$, denoted as $S \xrightarrow{e} S'$, where*

Function α'. *If the event e concerns activity instance $a = (activity(e), case(e))$ and a concludes, then the function α' is defined on the same domain as function α, except for a, and the function values are the same as those of α. Conversely, if e refers to an activity transition to a state different from conclusion, the function domain remains unchanged and the function value is only changed for activity instance a, i.e. $\alpha'(a) = state(e)$. More precisely:*

$$\alpha' = \begin{cases} \alpha \oplus \Big\{ \Big(\big(activity(e), case(e)\big), state(e) \Big) \Big\} & \text{if } state(e) \neq Concluded \\ \alpha \ominus \Big\{ \big(activity(e), case(e)\big) \Big\} & \text{if } state(e) = Concluded \end{cases}$$

Function ν'. *For each property $(name, val) \in properties(e)$, the value of the ν' function is obtained as follows: $\nu'(name, case(e)) = val$. For all other domain elements of ν', the function values are the same as those of function ν. More precisely:*

$$\nu' = \nu \oplus \big\{ \big((name, case(e)), val\big) \mid (name, val) \in properties(e) \big\}$$

Replaying is used to reconstruct the *Execution History*:

Definition 4 (Execution History). *Let $< e_1, \ldots, e_n >$ be the sequence of events in an execution log. Let $S_0 \xrightarrow{e_1} S_1 \xrightarrow{e_2} \ldots \xrightarrow{e_n} S_n$ be the sequences of states through which the system went. An* execution history *is a sequence of pairs $H =< (S_1, t_1), \ldots, (S_n, t_n) >$ where (S_i, t_i) denotes that the system entered state S_i at time $t_i = timestamp(e_i)$.*

4.3 Mapping States onto Maps

When replaying, the process moves from state $S_i = (\alpha_i, \nu_i)$ to state $S_{i+1} = (\alpha_{i+1}, \nu_{i+1})$ and these states need to be projected onto maps.

By leveraging on the map metaphor described in earlier sections, activity instances can be visualized by dots on the map. By not fixing the type of map, but allowing this choice to be configurable, different types of relationships can be shown thus providing a deeper insight into the context of the work to be performed. As mentioned before, a map can be a geographical map (e.g., the map of a university's campus), but other maps can also be used, e.g., process schemas, organizational diagrams, Gantt charts, etc.

An activity instance is represented as a dot positioned along certain coordinates on a background map. A map is meant to capture a particular perspective of the context of the process. Since an activity instance can be associated with several perspectives, it can be visualized on several maps at different positions. Maps can be designed at design time as needed. When the use of a certain map is envisaged, the relationship between activities and their position on the map should be specified through a function determined at design time.

Definition 5 (Position function). *Let A and V be the universe of the names of activities and process variables, respectively. Let U be the universe of the values of process variables. Let M be the set of maps of interest. For each available map $m \in M$, there exists a function*

$$position_m : A \nrightarrow Expr(V)$$

where $Expr(V)$ is the domain of all expressions that use some of the variables $v \in V$. Let $\xi : V \nrightarrow U$ be a value assignment of a subset of the variable names $v \in V$. We define $eval$ as a function which, given a position function f and a value assignment ξ, yields a pair of non-negative numbers:

$$eval[\![f]\!]\,(\xi) = (c1, c2)$$

where $(c1, c2) \in \mathbb{R}_0^+ \times \mathbb{R}_0^+$.

Given a map $m \in M$, a state $S_i = (\alpha_i, \nu_i)$, an activity instance $a = (a_{name}, a_{pid}) \in dom(\alpha_i)$ and a function $f_a = position_m(a)$, the coordinates of the position of activity instance a onto map m for state S_i is:

$$position_m(a)\big|_{S_i} = eval[\![f_a]\!]\,(\xi_a)$$

where $\xi_a(v_{name}) = \nu_i(v_{name}, a_{pid})$.

For a map $m \in M$, the function $position_m$ may be partial, since some elements of A may not have an associated position. It can also be the case that, for some activity $a \in dom(\alpha)$, $dom(\xi_a) \subset dom(position_m(a))$; in such a case, some variables of $f = position_m(a)$ have no assignment. Consequently, $eval[\![f]\!]\,(\xi)$ is undefined and, hence, a is not associated with any position on map m.

The projection of a state $S_i = (\alpha_i, \nu_i)$ onto a map m is the projection of each and every activity $a \in dom(\alpha_i)$ onto m at position $position_m(a)\big|_{S_i}$. The dot associated to activity a is also filled with a color that depends on $\alpha_i(a)$, i.e. the state of activity a. Table 4 summarizes the choice of colors used in the framework.

When dots overlap in a certain map, they are joined to form bigger dots, since otherwise some of them would be invisible. The diameters of such dots grow logarithmically with the number of activities amalgamated. The amalgamated dots are divided in as many slices as there are constituting dots and each slice is filled with the color of the dot that it corresponds to.

Section 4.4 will discuss how the different photographs capturing the various states can be merged, thus obtaining a movie. The remainder of this section summarizes the

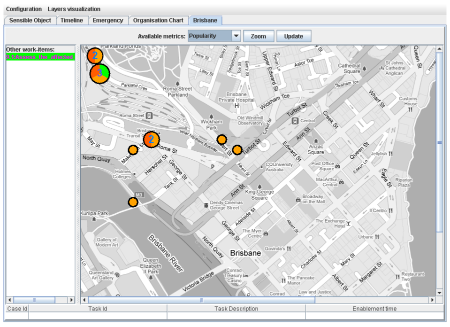
State	Associated Color
Unscheduled	Not applicable
Scheduled	White
Allocated	Cyan
Executing	Green
Suspended	Black
Concluded	Not applicable

Table 4. The color used to fill an activity dot depends on the state of the activity. The table shows the selection of colors that is envisioned in the framework. Activity instances that are unscheduled or concluded are not represented on a map.

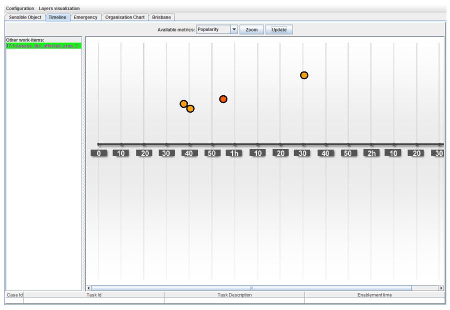
results of earlier work [12] that is based on a framework which aims to provide visual run-time support to process participants for the complex decision of choosing the next activity instance to work on. Although the framework is slightly different, we believe that its discussion can be of support to the readers to understand the framework described in this paper.

Visual Support for Activity Assignment In real scenarios, users can be confronted with a very large number of activity instances that stem from multiple cases of different processes. In this “jungle of work items”, users may find it hard to choose the right activity to work on next. The system cannot autonomously decide which is the right activity, since the decision is also dependent on conditions that are somehow “outside the system”. For instance, what is “best” for an organization should be mediated with what is “best” for its employees. Therefore, in our earlier work [12], a tool has been devised which provides an intuitive graphical interface that uses contextual information about activities and process participants to provide suggestions about prioritization of activity instances to perform. While the basic foundation is mostly identical, the earlier framework is slightly different from the one proposed in this paper. First, in [12] the state that is projected onto the map is always the current one. Moreover, different process participants are offered different sets of activity instances, since they may have different roles and privileges. Therefore, for a given participant, the projected activity instances are only those which the participant is allowed to view or to work on. In order to suggest the next activities to perform, the *scheduled* activities are not always colored white but with colors that range from white to black with different shades of yellow, orange, red and brown. Dots are colored according to a “distance notion”: different process participants may find the same activity closer or farther to their sphere, meaning they have different experiences with such an activity type, have different working speeds, and so on. A activity that is close tends to be colored black, meaning a participant is suggested to pick that as next; an activity that is far away is colored white, meaning the execution of that activity can be postponed.

Figure 11 shows two screenshots of the tool for two different maps. In particular, Figure 11(a) shows the projection of activity instances onto a geographic map. On this map, activity instances are placed at the physical location where they should be



(a) Projection of the current process state onto a geographic map.



(b) Projection of the current process state onto a timeline map.

Fig. 11. Using the map metaphor to distribute work. Dots show the current state on various maps used by end-users and managers.

executed. If the locations of some of them are so close that their corresponding dots overlap, these are amalgamated into a larger dot. The number inside such a larger dot corresponds to the number of activity instances represented by it. The larger dot is also divided in slices, one per amalgamated dot, and each slice is filled with the same color as the corresponding dot. Figure 11(b) illustrates the projection using a different map: activity instances are positioned onto the maps so that the x-axis position represents the time remaining before an activity instance expires, whereas the y-axis position is computed as a function of the case identifier, an positive integer.

4.4 Mapping Logs on Maps

In the previous subsections, we showed how to create an execution history H and a mapping of activity instance states onto a map $m \in M$ based on function $position_m$. Recall that the execution history is sequence of states and corresponding timestamps:

$$H =< (S_1, t_1), \ldots, (S_n, t_n) >$$

At time t_i the i^{th} event causes the process to move to state $S_i = (\alpha_i, \nu_i)$. The challenge is now to map such a sequence of states onto a map. A state corresponds to a photograph and an execution history corresponds to a movie. Process instances can run for days, weeks, or even months. There may be many events per process instance. Moreover, activities may take a very short time compared to the overall flow time of a process instance. Therefore, a naïve approach that simply concatenates photographs into a movie is destined to fail. Some of the complications are:

– The number of events may be too large to visualize in a movie due to performance reasons or visualization problems (cluttered diagrams).

– The speed of the movie should correspond to real time rather than the number of events. For example, a day with just a few events should take the same time as a day with many events. (Unless the user deliberately wants to use logical timestamps rather than real timestamps.)
– The duration of activities may be very short compared to the flow time of process instances. For example, an activity may be atomic (takes no time according to the event log) or takes just a few minutes in a process that has an average flow time of weeks. Note that an activity of one minute corresponds to 0.00992 percent of the flow time of a process taking a week. Hence, a naïve visualization approach would make such event invisible.

To address these problems we select a refresh rate $1/\tau$ and build an aggregate picture for each period of τ time units. This is similar to a television using NTSC or PAL. According to the NTSC standard the image is refreshed 60 times per second (60 Hz). PAL uses a refresh rate of 50 Hz. Figure 12 shows the basic idea. The time period covered by the event log is partitioned into smaller periods of τ time units. Per τ period, there is a collection of events, resulting in a sequence of states. The photographs of these states are aggregated into a single photograph. The aggregated photographs of all time periods are concatenated into a movie. Given an event log, different refresh rates can be used to create a more fine-grained movie or a more coarse-grained movie.

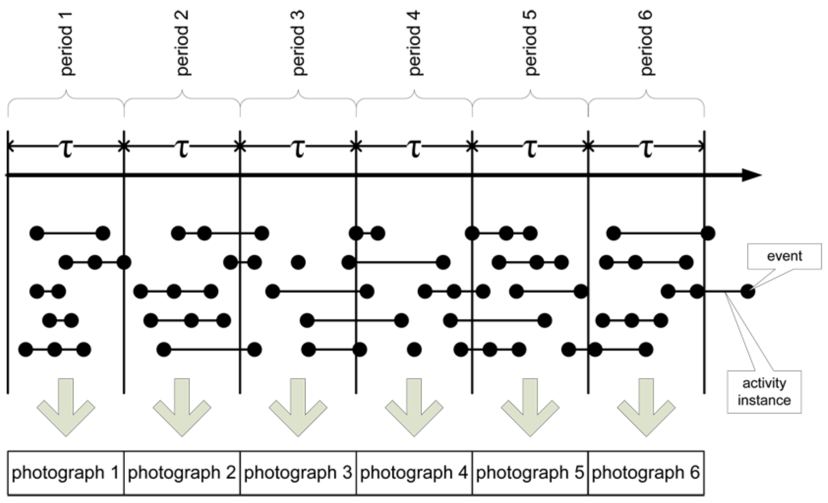
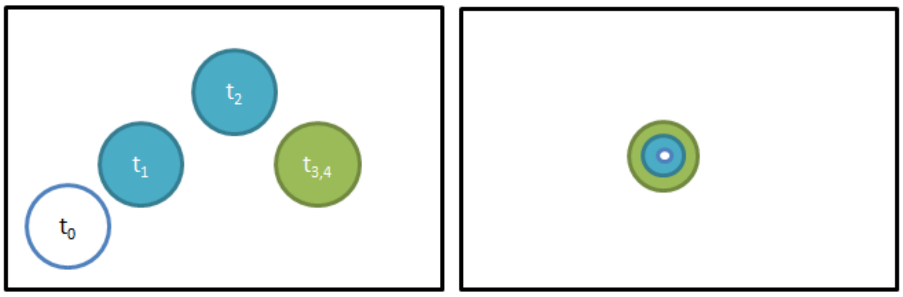


Fig. 12. Creating movies based on an event log using a refresh rate of $1/\tau$; one photograph is created for every period of τ time units.

The approach depicted in Figure 12 helps us to address the problems mentioned before. First of all, the aggregation per period can be used to avoid cluttering the map and to improve performance for huge data sets. Second, by showing all aggregated images equally long, the speed of the movie automatically corresponds to the real time progression. Third, through aggregation and scaling, the visualization can also show activities that run for a short period compared to the overall flow time. Even when activities take only a fraction of the flow time, the depicted level of activity can be shown relative to other periods.

Again we use the notion of dots having a position, color, and diameter. The position of a dot on a map depends on the properties of the corresponding events. Smaller dots can be amalgamated into larger dots. The color and size of such amalgamated dots may depend on various properties of the underlying events (frequency, duration, etc.).



(a) Projection of an activity as a dot in six subsequent system states $\overline{S_0}, \overline{S_1}, \overline{S_2}, \overline{S_3}, \overline{S_4}, \overline{S_5}$ of an execution history H. For the photograph of $\overline{S_5}$, the activity was concluded and, hence, the associated dot is no longer visualized.

(b) Projection of the activity as a dot in a composite photograph that merges the photographs of system states $\overline{S_0}, \overline{S_1}, \overline{S_2}, \overline{S_3}, \overline{S_4}, \overline{S_5}$.

Fig. 13. Example of computing the position, the radius and the colors of a dot in a composite photograph.

Figure 13 shows an example focusing on a particular activity instance that goes through a sequence of states. The photograph aggregates six states $\overline{S_0}, \overline{S_1}, \overline{S_2}, \overline{S_3}, \overline{S_4}, \overline{S_5}$ of the history $H =< \ldots, (\overline{S_0}, \overline{t_0}), \ldots, (\overline{S_5}, \overline{t_5}), \ldots >$. Figure 13(a) illustrates the projection of an activity as a dot in the photographs for system states $\overline{S_0}, \overline{S_1}, \overline{S_2}, \overline{S_3}, \overline{S_4}$. In state $\overline{S_5}$ the activity instance was concluded and, hence, the associated dot is no longer visualized. In the photographs associated to the different states, the dots are colored differently, since the activity instance is in different states. Figure 13(b) shows the visualization of the activity-instance dot in the aggregated photograph for the time period. The dot's position is computed as the average of the positions of the dots in the original photographs. The average is weighted with respect to the durations the dot was located at the various positions in the time period. The dot area is split in five rings: one is white colored, two are cyan, and two are green since the activity instance was, respectively, in the scheduled state for system state $\overline{S_0}$, in the allocated state for $\overline{S_1}$ and $\overline{S_2}$ and in state executing for $\overline{S_3}$ and $\overline{S_4}$ (see Table 4).

The radius of each ring is directly proportional to the duration the activity instance was in the state corresponding to that ring

5 Tool Support

The process mining framework ProM was developed based on experiences with simple process mining tools such as *MiMo* (α-miner based on ExSpect), *EMiT* (α-miner taking transactional information into account), *Little Thumb* (predecessor of the heuristic miner), *InWolvE* (miner based on stochastic activity graphs), and *Process Miner* (miner assuming structured models) [13]. To avoid "reinventing the wheel" each time a new process mining algorithm was developed, researchers at Eindhoven University of Technology developed ProM, a "plug-able" environment for process mining using MXML (Mining XML format) as input format. The goal of the first version of this framework was to provide a common basis for all kinds of process mining techniques, e.g., supporting the loading and filtering of event logs and the visualization of results. This way people developing new process discovery algorithms did not have to worry about extracting, converting, and loading event data. Moreover, for standard model types such as Petri nets, EPCs, and social networks default visualizations were provided by the framework. In 2004, the first fully functional version of ProM framework (*ProM 1.1*) was released. This version contained 29 plug-ins. With each version of ProM the number of plug-ins increased. For example, *ProM 5.2* was released in 2009 and contained 286 plug-ins: 47 mining plug-ins, 96 analysis plug-ins, 22 import plug-ins, 45 export plug-ins, 44 conversion plug-ins, and 32 filter plug-ins. Note that the Fuzzy miner is just one of these 286 plug-ins. The spectacular growth of the number of plug-ins in the period from 2004 to 2009 illustrates that ProM realized its initial goal to provide a platform for the development of new process mining techniques. ProM has become the de facto standard for process mining. Research groups from all over the globe contributed to the development of ProM and thousands of organizations downloaded ProM.

ProM 6 (released in November 2010) is based on XES rather than MXML. XES is the new process mining standard adopted by the IEEE Task Force on Process Mining. Although ProM 5.2 was already able to load enormous event logs, scalability and efficiency were further improved by using OpenXES. ProM 6 can distribute the execution of plug-ins over multiple computers. This can be used to improve performance (e.g., using grid computing) and to offer ProM as a service. The user interface has been re-implemented to be able to deal with many plug-ins, logs, and models at the same time. Plug-ins are now distributed over so-called packages and can be chained into composite plug-ins. Packages contain related sets of plug-ins. ProM 6 provides a so-called package manager to add, remove, and update packages. Users should only load packages that are relevant for the tasks they want to perform. This way it is possible to avoid overloading the user with irrelevant functionality. Moreover, ProM 6 can be customized for domain specific or even organization specific applications.

Figure 14 shows the architecture of the implementation of the framework to provide visual spatio-temporal analysis of process executions as described in Section 4. In particular, the framework is being implemented as a ProM plug-in. The plug-in requires two input objects:

The Event Log. The XML file of the event log in XES format. This event log is assumed to be produced by some information system. The assumption of using event logs in this format is not a strong assumption since ProM is provided with a number of different import plug-ins that can convert execution logs from many formats to XES. Moreover, ProM is highly extensible and, hence, it is possible to extend it with new import plug-ins to load logs in whichever format.

Map Specifications & Specification of the Projection of Activity Instances on maps. This is an additional XML file which includes crucial information. In particular, such an XML file codes the set M of available maps (i.e. the map name and the URL where the map image can be retrieved) as well as the definition of the *position* function (Definition 5) for each map.

When the plug-in is launched with the required input objects, the user is asked to specify the value for the time-slot period τ in order to define the level of aggregation initially requested. The result of the plug-in application is a set of movies, one for each map. Users can play the movie, move forward or backward, or jump to a certain time slot of interest. Moreover, in order to drill down or roll up, user can also zoom in or out of a movie and change the time-slot period in order to split a composite photograph into a set of them or to merge several into a single one. Note that the implementation of the plug-in is ongoing as part of a Master project at TU/e.

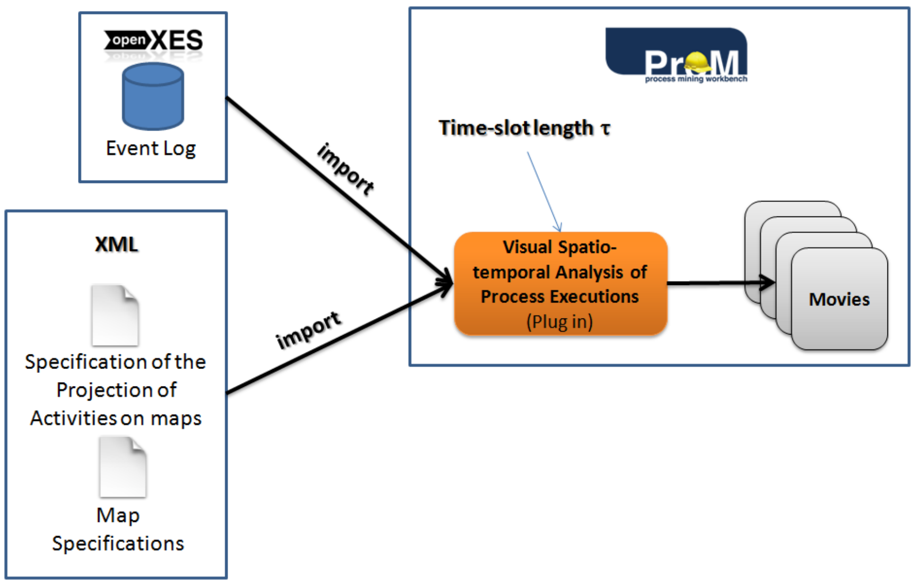


Fig. 14. Architecture of the implementation of the framework described in Section 4 as ProM plug-in

6 Related Work

In this paper, we proposed to combine process mining and visual analytics. Both are emerging domains that attracted considerable attention in the last couple of years. Interested readers can see [13] for an overview of the early work on process mining, whereas the book [1] provides a more comprehensive overview of the state-of-the-art on process mining.

In the past, many automatic techniques for process mining, but also for statistics and data mining, have been developed without taking the visualization aspects into account. On the other hand, the research community focusing on information visualization (e.g. [14] or [15]) did not address process-related issues like in the process mining domain.

For what concerns visualizations in the field of business processes, the work described in [16] allows the current status of the running business process instances to be visualized in 2D diagrams. These diagrams merge different viewpoints, as our framework aims to do. Unfortunately, the admissible diagrams are restricted to traditional bar charts and histograms meshed to process schemas, which are visualized as undirected graphs. Therefore, the approach is not fully flexible, since users cannot define specific domain-dependent maps that may give an insight into relevant perspectives for a certain class of processes, unless they are expressed as bar charts or histograms. The research work reported in [17] builds on [16] and, additionally, provides a 3D business process visualization. It relies on the concept of 3D Gadgets, which are a set of graphical primitives that can be combined to define relevant diagrams. However, the possible alternative diagrams that can be obtained are quite restricted, since the number of available gadgets is limited and new ones cannot be defined by users.

Also related is work on business process mash-ups, which is a technique for building applications that combine information from multiple sources with process data to create an integrated view. Nowadays, there is not a complete framework to mash up process data with those of external sources as this is non-trivial: it would be necessary to agree upon standard ontologies that need to be integrated with the process data perspective. Hence contemporary process mash-up solutions are still ad-hoc, developed either for specific process specifications or for certain data sources. For instance, for mashing up geographical information and process data, there exist several research papers [18, 19] and even commercial products.[3] Combining geographical information with business processes is only one of the usages that we allow for. The framework proposed in this paper is not restricted to mere geographical data: maps can be geographical but also other map types are allowed and for non-geographical maps the concept of activity locations takes on a completely different meaning.

The map metaphor has been widely used to provide user-friendly Visual Query Languages to retrieve data from databases [20, 21]. In particular, Chang [20] proposes the concept of *sentient maps*, which may be of any nature (e.g., geographic, documental) and are conceptually similar even though applied to querying databases.

Of course, *visual analytics* is more than only visualization: it is an integrated approach that tries to incorporate human perceptiveness into techniques techniques that

[3] E.g., Kinetic by linkuistics `http://www.linkuistics.com.au/Products`

combine visualization and data analysis. A first attempt of merging these aspects has successfully been conducted for data mining [22].

The term "visual analytics" was coined in [3], which reviews the early work in this field. An up-to-date and comprehensive reference for readers interested in the topic is [5]. Examples of recent significant research in the area of visual analytics can be found in document analysis [23], financial analysis [24, 25] and geo-spatial object analysis [26]. Another interesting example of the application of visual analytics is provided in Willems et al. [27], where a technique is proposed to visually summarize the movements of vessels along the Dutch coast in a single display.

Obviously, the scope of visual analytics does not exclude the visualization of process-related data. However, most of the work is either quite generic or quite specific for a particular application. What is missing is a systematic approach to make visual analytics "process aware". Temporal aspects, concurrency, causality etc. are process aspects that need to be visualized in a consistent and systematic manner.

7 Conclusion

Process mining is an emerging discipline that analyzes event data from a process-oriented perspective. Today's process mining techniques are able to extract important insights from semi-structured processes. Visual analytics focuses on visualizing information in a simple and human-understandable way, thereby using the amazing capabilities of humans to see patterns in unstructured data. As advocated in this paper, process mining and visualization can be combined to maximize their effectiveness.

We positioned data mining, process mining, visualization and visual analytics and showed that a combination of approaches is needed. To do this we used the metaphor of a map. Process models can be seen as maps and ideas from cartography can be used to make these models better understandable. Moreover, process models are just one view on processes and, depending on the questions one seeks to answer, an array of maps should be provided. The current state of a process can be projected onto such maps, thus visualizing a "photograph of the process". As shown, individual photographs can be aggregated into images that can be put into sequence and thus form a "process movie". Earlier experiments with the Fuzzy miner in ProM and work assignment in the context of YAWL, show that such an approach is feasible and effective. However, these techniques should only be seen as the starting point of a more comprehensive approach to "breathe life" into otherwise static business process models.

Future research focuses on two main directions. First of all, we are implementing the framework described in Section 4. As indicated in Section 5, we expect to operationalize the approach in terms of a ProM plug-in. Second, we want to make use of visual-analytics techniques in the field of operational support [1, 28] to guide users in executing activities in business processes. Operational support is a set of techniques to recommend users about the next actions to perform for reaching certain desired goals without violating business and operative constraints. Usually, there exist multiple applications, referred to as "operational support providers", that provide recommendations to process participants. In [12] we used maps to guide users in selecting work items. However, this is only one of many possible applications. We plan to provide comprehensive support for this in the context of ProM [1].

References

1. Aalst, W.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer-Verlag, Berlin (2011)
2. Rozinat, A., Aalst, W.: Conformance Checking of Processes Based on Monitoring Real Behavior. Information Systems **33** (2008) 64–95
3. Thomas, J., Cook, K., eds.: Illuminating the Path: The Research and Development Agenda for Visual Analytics, IEEE CS Press (2005)
4. Keim, D., Mansmann, F., Schneidewind, J., Thomas, J., Ziegler, H.: Visual analytics: Scope and challenges. In Simoff, S.J., Böhlen, M.H., Mazeika, A., eds.: Visual Data Mining. Springer-Verlag (2008) 76–90
5. Keim, D., Kohlhammer, J., Ellis, G., Mansmann, F., eds.: Mastering the Information Age: Solving Problems with Visual Analytics, VisMaster, http://www.vismaster.eu/book/ (2010)
6. Günther, C., Aalst, W.: Fuzzy Mining: Adaptive Process Simplification Based on Multi-perspective Metrics. In Alonso, G., Dadam, P., Rosemann, M., eds.: International Conference on Business Process Management (BPM 2007). Volume 4714 of Lecture Notes in Computer Science., Springer-Verlag, Berlin (2007) 328–343
7. Aalst, W., Weijters, A., Maruster, L.: Workflow Mining: Discovering Process Models from Event Logs. IEEE Transactions on Knowledge and Data Engineering **16** (2004) 1128–1142
8. Weijters, A., Aalst, W., Medeiros, A.: Process Mining with the Heuristics Miner-algorithm. BETA Working Paper Series, WP 166, Eindhoven University of Technology, Eindhoven, The Netherlands (2006)
9. Medeiros, A., Weijters, A., Aalst, W.: Genetic Process Mining: An Experimental Evaluation. Data Mining and Knowledge Discovery **14** (2007) 245–304
10. Song, M., Aalst, W.: Towards Comprehensive Support for Organizational Mining. Decision Support Systems **46** (2008) 300–317
11. Günther, C.: Process Mining in Flexible Environments. PhD Thesis, Eindhoven University of Technology, Eindhoven, The Netherlands (2009)
12. Leoni, M., Aalst, W., Hofstede, A.: Visual Support for Work Assignment in Process-Aware Information Systems. In Dumas, M., Reichert, M., Shan, M., eds.: International Conference on Business Process Management (BPM 2008). Volume 5240 of Lecture Notes in Computer Science., Springer-Verlag, Berlin (2008) 67–83
13. Aalst, W., Dongen, B., Herbst, J., Maruster, L., Schimm, G., Weijters, A.: Workflow Mining: A Survey of Issues and Approaches. Data and Knowledge Engineering **47** (2003) 237–267
14. Chen, C.: Information Visualization: Beyond the Horizon. Springer-Verlag, New York, Inc. (2006)
15. Spence, R.: Information Visualization: Design for Interaction. 2nd edn. Pearson Education Limited, Harlow, England (2006)
16. Schönhage, B., Eliëns, A.: Management Through Vision: A Case Study Towards Requirements of BizViz. In: AVI 2000: Internation Conference of Information Visualisation, IEEE Computer Society (2000) 387–392
17. Schönhage, B., van Ballegooij, A., Elliëns, A.: 3D Gadgets for Business Process Visualization—A Case Study. In: VRML '00: Proceedings of the fifth symposium on Virtual reality modeling language (Web3D-VRML), ACM (2000) 131–138
18. Alonso, G., Hagen, C.: Geo-Opera: Workflow Concepts for Spatial Processes. In: SSD'97: Proceedings of the 5th International Symposium on Advances in Spatial Databases. Volume 1262 of Lecture Notes in Computer Science., Springer (1997) 238–258

19. Kaster, D., Bauzer-Medeiros, C., da Rocha, H.V.: Supporting Modeling and Problem Solving from Precedent Experiences: The Role of Workflows and Case-based Reasoning. Environmental Modelling and Software **20** (2005) 689–704
20. Chang, S.: The Sentient Map. Journal of Visual Language and Computing **11** (2000) 455–474
21. Camara, K., Jungert, E.: A Visual Query Language for Dynamic Processes Applied to a Scenario Driven Environment. Journal of Visual Languange and Computing **18** (2007) 315–338
22. Keim, D.: Visual Exploration of Large Data Sets. Communications of the ACM **44** (2001) 38–44
23. Oelke, D., Ming, C., Rohrdantz, C., Keim, D., Dayal, U., Lars-Erik, H., Janetzko, H.: Visual Opinion Analysis of Customer Feedback Data. In: Proceedings of the IEEE Symposium on Visual Analytics Science and Technology (IEEE VAST 2009), IEEE (2009) 187–194
24. Keim, D., Nietzschmann, T., Schelwies, N., Schneidewind, J., Schreck, T., Ziegler, H.: A Spectral Visualization System for Analyzing Financial Time Series Data. In: Proceedings of the Eurographics/IEEE-VGTC Symposium on Visualization (EuroVis 2006), Eurographics Association (2006) 195–200
25. Ziegler, H., Nietzschmann, T., Keim, D.: Relevance Driven Visualization of Financial Performance Measures. In: Proceedings of the Eurographics/IEEE-VGTC Symposium on Visualization (EuroVis 2007), Eurographics Association (2007) 19–26
26. Bak, P., Mansmann, F., Janetzko, H., Keim, D.: Spatio-temporal Analysis of Sensor Logs using Growth Ring Maps. IEEE Transactions on Visualization and Computer Graphics **15** (2009) 913–920
27. Willems, N., van de Wetering, H., van Wijk, J.: Visualization of Vessel Movements. In: Proceedings of the Eurographics/IEEE-VGTC Symposium on Visualization (EuroVis 2009), Eurographics Association (2009) 959–966
28. Rozinat, A., Wynn, M., Aalst, W., Hofstede, A., Fidge, C.: Workflow Simulation for Operational Decision Support. Data and Knowledge Engineering **68** (2009) 834–850