Softw Syst Model (2016) 15:707–732
DOI 10.1007/s10270-014-0432-2

REGULAR PAPER

Turning event logs into process movies: animating what has really happened

Massimiliano de Leoni · Suriadi Suriadi · Arthur H. M. ter Hofstede · Wil M. P. van der Aalst

Received: 18 December 2013 / Revised: 8 August 2014 / Accepted: 15 August 2014 / Published online: 28 September 2014

Abstract Today's information systems log vast amounts of data. These collections of data (implicitly) describe events (e.g. placing an order or taking a blood test) and, hence, provide information on the actual execution of business processes. The analysis of such data provides an excellent starting point for business process improvement. This is the realm of process mining, an area which has provided a repertoire of many analysis techniques. Despite the impressive capabilities of existing process mining algorithms, dealing with the abundance of data recorded by contemporary systems and devices remains a challenge. Of particular importance is the capability to guide the meaningful interpretation of "oceans of data" by process analysts. To this end, insights from the field of visual analytics can be leveraged. This article proposes an approach where process states are reconstructed from event logs and visualised in succession, leading to an animated history of a process. This approach is customisable in how a process state, partially defined through a collection of activity instances, is visualised: one can select a map and specify a projection of events on this map based on the properties of the events. This paper describes a comprehensive implementation of the proposal. It was realised using the open-source process mining framework ProM. Moreover, this paper also reports on an evaluation of the approach conducted with Suncorp, one of Australia's largest insurance companies.

Keywords Process mining · Visual analytics · Event-log animation · Process visualisation

Communicated by Dr. Jordi Cabot.

M. de Leoni (✉) · A. H. M. ter Hofstede · W. M. P. van der Aalst
Eindhoven University of Technology, Eindhoven, The Netherlands
e-mail: m.d.leoni@tue.nl

A. H. M. ter Hofstede
e-mail: a.terhofstede@qut.edu.au

W. M. P. van der Aalst
e-mail: w.m.p.v.d.aalst@tue.nl

S. Suriadi · A. H. M. ter Hofstede · W. M. P. van der Aalst
Queensland University of Technology, Brisbane, Australia
e-mail: s.suriadi@qut.edu.au

1 Introduction

Process-aware information systems (PAIS) are increasingly used by organisations to support their businesses. All these systems record the execution of process instances in so-called event logs. These logs thus capture information about activities performed. Each event records the execution of an activity instance by a given resource at a certain point in time along with the output produced.

Event logs are the key enabler for *process mining* [32] (a field of study that is capable of extracting, from event logs, in-depth insights into process-related problems that contemporary enterprises face). Through the application of process mining, organisations can discover the processes as they were conducted in reality, check whether certain practices and regulations were really followed and gain insight into bottlenecks, resource utilisation, and other performance-related aspects of processes.

The amount of process data that today's enterprises produce is growing exponentially. For example, the McKinsey Global Institute (MGI) estimated that globally, in 2010, enterprises stored more than 7 exabytes of new data on disk drives, most of which is related to the execution of their process [22]. Such a large availability of new process data makes it possible to apply various forms of process mining analyses. Nonetheless, the sheer quantity and diversity of these data, unfortunately, creates new challenges [19]. Dozens of process min-

ing techniques are available to process analysts for different purposes, such as for compliance checking, bottleneck analysis, and resource-utilisation analysis. However, they become far less effective if process analysts do not know which aspects or which parts of the data are worthy of investigation. In this "ocean of data", process analysts need to be guided with regard to where to focus their analysis efforts. For example, if activities/process instances are always executed on time, there is no reason to waste time to verify the cause of bottlenecks, as they are not present. Similarly, if resources are usually not overloaded, it is not worthwhile to perform an analysis of their utilisation.

The starting point of this article is the belief that, by visually replaying event logs, process analysts can gain insights into potential problems that can later be confirmed or refuted through the execution of other process mining analyses. Existing approaches are unable to provide a visual and dynamic representation of what has happened while process instances are being executed. This paper proposes a technique to provide process analysts with *a tool to visually replay the behaviour of executed process instances as recorded in the event log*. More importantly, during replay, the log can be represented in various customisable forms to suit different needs. Visually replaying event logs can also be useful when tuning the application of process mining techniques, e.g. when setting the parameters of such techniques.

The visualisation framework proposed in this article leverages on ideas from *Visual Analytics*, a term coined by Jim Thomas in [31]. We can define visual analytics as an approach, which *combines automated analysis techniques with interactive visualisations for an effective understanding, reasoning, and decision making on the basis of very large and complex data sets* [16].

To our knowledge, little work exists that combines visual analytics with process mining for the purpose of visualising event logs in a dynamic manner. On the one hand, process mining research often tends to overlook visualisation aspects. On the other hand, research in the domain of visual analytics does not put enough emphasis on providing support to visualise process-related aspects, notably the time dimension. A more detailed discussion in the current research gap is provided in Sect. 6.

The framework is based on the concept of *process-oriented maps* that are simply called "maps" henceforth. A *map* encodes a different viewpoint on the execution of process instances, such as geographical distribution of work, time aspects (e.g. possible deadlines), process structure, resource involvement (e.g. the company's organisation), or data-related characteristics of process instances. A map can be handcrafted using different types of diagrams, such as a process model, a Gantt chart, a geographical map, and an organisational chart. At the same time, as detailed in Sect. 3, our implementation also supports automatic generation of a few types of maps, such as a timeline map and a Cartesian graph-based map.

Activities as recorded in an event log can be projected as dots onto such maps in meaningful positions. For example, executed activities as seen in an event log can be projected onto a process model map, positioned on the node corresponding to the respective activity types. As an alternative, executed activities can be positioned onto the geographical locations where they were being performed. Furthermore, activity instances can be visualised on a timeline map according to the due deadline.

At the same time, dots can be coloured and the colour can be determined by some customisable process characteristics, such as the age of the activity instance, its state, or the type of process instances to which the activity instance belongs (e.g. referring to gold or silver customers or to large or small insurance claims). For example, dots can be coloured differently depending on the states of the activity instances they represent at any particular point in time (such as being executed, suspended, or offered for performance to a resource or group of resources). If multiple activity instances are projected as dots onto the same position, the dots are merged so as to produce a pie chart with multiple slices. The pie chart is divided into as many slices as the number of merged dots. Each slice is associated with a different merged dot and coloured accordingly. Also, the pie-chart diameter grows with the number of dots merged. If a dot is not overlapping with any other, one can still see it as a pie chart with a single slice. That is the reason why, in the remainder, *we are going to use the term pie chart, even if only one activity instance is involved*.

Using event-log information, it is possible to replay the history and build the states the system went through. Hence, for each "map", a sequence of different "photographs" can be built, showing how activities were projected on the map in each of these states. If, for each map, the constructed sequence of photographs is played in succession, one obtains a different "process movie". These movies or animations (one per map) provide analysts and domain experts with a helicopter view of the past execution history seen from different perspectives.

The implementation was realised through two plug-ins to the generic open-source process mining tool, called the ProM tool,[1] as well as a stand-alone application in Java.

We conducted a pilot study with stakeholders from a Dutch municipality to assess the usefulness of the tool in aiding users to gain relevant insights. As a result of the feedback, the framework and the corresponding implementation was improved to cope with the problems that were identified. Afterwards, another evaluation effort was made, this time with subjects from Suncorp, an Australian insurance company. The Suncorp use case focussed on the process of

[1] http://www.promtools.org/.

claims handling from policy holders who want to receive a reimbursement for home-related damages. In both evaluations, the main goal was to assess whether the map metaphor could be understood by users and whether the use of movies allowed users to gain meaningful insights into their processes. As detailed later in this article, we obtained convincing positive feedback.

The paper is organised as follows. Section 2 discusses the visualisation framework. Then, Sect. 3 provides some details of the implementation of this framework realised as a plug-in of the ProM environment. Section 4 describes a case study, illustrating the approach in the context of insurance claims processing for claims related to home damages at Suncorp. Section 5 starts with a summary of the first evaluation of our approach with subjects from a Dutch municipality. Then, the section continues with the details of the evaluation of the improved framework in collaboration with subjects from Suncorp. Section 6 positions our work with respect to the literature, highlighting existing limitations. In particular, we review the literature on visualisation in the field of business process management and business process intelligence, and we discuss their intersection with the field of visual analytics. Section 7 concludes the paper by summarising the results obtained and by identifying possible avenues for future work.

2 Background and framework

This section introduces a formal framework that allows event logs to be visualised and replayed on generic maps, thus obtaining "movies".

These movies aim to facilitate the analysis of the history of the execution of process instances as recorded in an event log. As historical data can be analysed from different angles, each movie is intended to enable process analysts to focus the analysis on a particular view.

The input of the framework is an event log and a set of maps of interest. The output is a set of "movies", one for each map. The events in an event log are sorted chronologically and are subsequently replayed in that order. The occurrence of an event makes the system enter a particular state. Hence, by replaying all events in the event log in chronological order, it is possible to rebuild a process history, i.e. the sequence of states the system went through. Each state can be represented as a set of pie charts projected onto a map. Each pie chart captures one or more activity instances of the same type belonging to different process instances.

In order to define how states can be represented on a map, a process expert needs to choose an image for that map and to define the position of activity instances as pie charts on that image. Such an annotated map can be seen as a "photograph", and thus, a process history can be visualised as a sequence of such photographs, which together form a "movie". To convey more information, the slices in pie charts can be filled with colours, where the colouring may depend on the characteristic of interest.

While the implementation is discussed later in detail, we already show a screenshot of our tool to provide a more concrete understanding of the visualisation framework proposed in this article. Figure 1 shows a "photograph" of the activity instances at a particular point in time (here, "23 Jan 2012 10:11:02") for the Suncorp use case. The screenshot shows a photograph of a movie of a geographical nature. Insurance claims that were still being processed are projected onto the Australian state of the claimants. Here, each activity instance is projected onto the map of Australia as a pie chart. An activity instance is shown in a pie chart for the duration that it was being executed as recorded in the log. The geographical states of claimants determine the position of the pie charts on the map. At any given point in time, the size of the pie charts associated with a particular state in Australia represents the number of activity instances that refer to the claims of that state (e.g. Queensland). The activity instances can have different characteristics. Therefore, the pie charts are sliced according to the percentage of instances with given characteristics, with different colours assigned to the slices corresponding to these characteristics (in Fig. 1, they are sliced according to claims' type, such as fire and theft). The time slider at the bottom of the figure allows users to quickly go back and forth to a certain point in time to view the corresponding photograph. Alternatively, the photographs can be played in a continuous sequence by clicking/selecting the "play" button.

Of course, the map shown in Fig. 1 is just one possible map supported by our framework. As detailed later in this article, an unlimited number of maps can be used, of which process model maps, timeline maps, and Cartesian graph-based maps are just examples.

Section 2.1 provides the background to properly formalise our framework. Section 2.2 introduces the formalisation of system states and transitions between two states that suits for the purpose of creating a movie. Section 2.3 formalises the projection of system states onto a map to create a set of photographs and subsequently a movie. Sections 2.4 and 2.5 explain additional features of our framework with regard to colouring pie charts and merging pie charts when multiple pie charts are projected onto overlapping positions.

2.1 Background

The background knowledge introduced here is compliant with the definition of Extensible Event Stream (XES),[2] which was adopted as a standard by the *IEEE Task Force on Process*

[2] http://www.xes-standard.org.

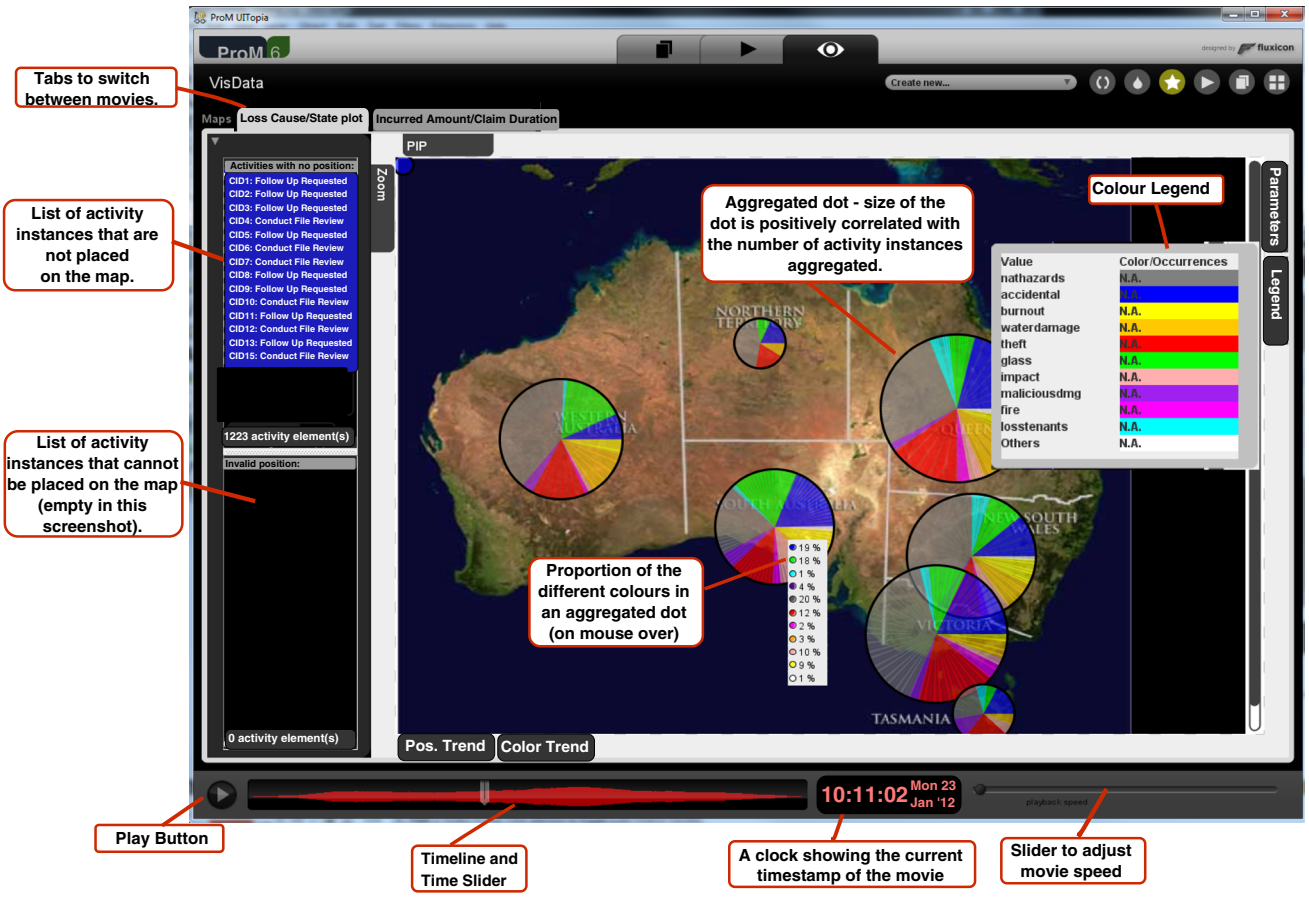


Fig. 1 A screenshot of a process movie referring to the log data of a process enacted in an Australian insurance company to deal with claims. *Open claims* are projected onto the Australian state of the claimants. The *pie charts* are sliced according to the proportion of the different types of claims (e.g. due to natural hazards, thefts, or fires). The image has been slightly edited from the original screenshot to improve readability

Mining[3] to represent logging data related to process events, often referred to as event logs.

Our framework does not require any assumptions about the process structure. In particular, as it will become clear later, we do not need an explicit representation of the dynamics of processes, such as the flow and dependencies between activities. Our framework only requires one to know the set of activity types (denoted as A) in the process being visualised and the names of process instance variables, i.e. the data objects, that are manipulated by process instances (denoted as V). This information can easily be extracted from event logs, even when the precise process is unknown.

In the remainder of this article, we use the word **case** to refer to *process instance*. We denote the set of case identifiers with C.

Definition 1 (*Activity Instance*) An *activity instance* is the execution of a certain activity in a certain case. It is identified as a pair (at, cid) where $at \in A$ is the activity type and $cid \in C$ is the case identifier.

A case manipulates a number of case variables.

Definition 2 (*Case Variable*) A *case variable* is a pair (vn, cid) where $vn \in V$ is the variable name and $cid \in C$ is the case identifier.

These variables can take on different values in different cases and also within the same case as time progresses.

Activity instances can be in a number of states. We use the notation Z to denote the set of possible states of instances, i.e. $Z = \{Scheduled, Assigned, Executing, Suspended, Concluded\}$:

- *Scheduled*: activity instance was created but was not yet assigned to a resource;
- *Assigned*: activity instance was assigned to a resource but not yet started;
- *Executing*: activity instance had commenced;
- *Suspended*: activity instance was temporarily halted with the possibility of continuing the execution later;
- *Concluded*: activity instance was completed.

[3] http://www.win.tue.nl/ieeetfpm.

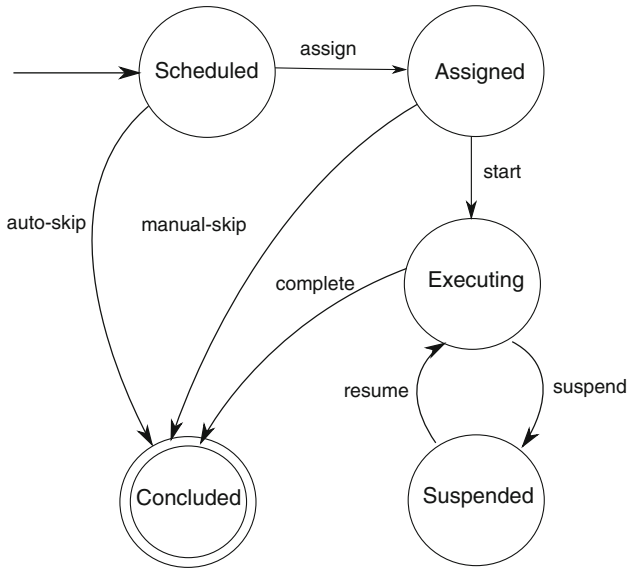


Fig. 2 Life cycle of activity instances

The life cycle of an activity instance is as in Fig. 2. In fact, this information can easily be deduced from an event log. An activity instance is referred to as *active* if it is in any state in Z except *Concluded*. These states and the valid transitions are compliant with the XES standard mentioned before.

Our visualisation framework is about replaying event logs. An event log consists of sequences of events.

Definition 3 (*Event*) Let T be the universe of possible timestamps. Let U be the universe of values that variables can take. An *event* e is a tuple $e = (at, cid, t, z, P)$ where

- $(at, cid) \in A \times C$ is an activity instance;
- $t \in T$ is the timestamp when event e occurred;
- $z \in Z$ is the state to which the corresponding activity instance moves;
- $P : V \nrightarrow U$ is an assignment of values to variables.[4] Function P is partial since some events may not provide a value for all process variables.

We use the following functions to access the constituent elements of an event $e = (at, cid, t, z, P)$: $activity(e) = at$, $case(e) = cid$, $timestamp(e) = t$, $state(e) = z$, and $properties(e) = P$. The latter function will be overloaded such that $properties(e, vn) = P(vn)$. Moreover, given a function f, $dom(f)$ represents the domain of f.

[4] We use $\nrightarrow$ to denote that a function is partial.

2.2 Creation of the sequence of states

Every time an event occurs, the system state changes. The framework is concerned with showing the evolution of the system state by projecting the sequence of states that the system has gone through on maps.

For the purpose of projection on maps, a system state is formalised as follows:

Definition 4 (*System State*) A system state $S = (\alpha, \nu, \tau^s)$ consists of

- a function $\alpha : (A \times C) \nrightarrow Z$ where $\alpha(at, cid) = z$ denotes that activity instance (at, cid) is in state z;
- a function $\nu : (V \times C) \nrightarrow U$ where $\nu(vn, cid) = v_{value}$ denotes that variable (vn, cid) has value v_{value};
- a function $\tau^s : (A \times C) \nrightarrow T$ where $\tau^s(at, cid) = t$ denotes that activity instance (at, cid) started at timestamp t.

These functions are defined as partial since some activity instances (at, cid) may have never been scheduled for execution and, hence, started. Similarly, some variables vn may have never taken on a value in a process instance cid.

Similar to existing algorithms for conformance checking [32], this framework is based on the principle of replay. Events in the log are replayed to determine, a posteriori, the sequence of states that the system went through.

The sequence of states is built iteratively by replaying all events in the log. As an activity instance enters the system, we register its state. At each iteration, the earliest not-yet-replayed event e is considered, thus triggering a system transit from a state S_i to a new state S_j.

The new state is constructed by overriding the functions α, ν, and τ associated with S_i to obtain the new functions to associate with S_j. Therefore, before formally defining how a system transits from one state to another, we need to introduce the overriding operators $\oplus$ and $\ominus$. Let f be a function, a function $f' = f \oplus (\overrightarrow{x}, y)$ is defined by $f'(\overrightarrow{x}) = y$ and $f'(\overrightarrow{x'}) = f(\overrightarrow{x'})$ for all $\overrightarrow{x'} \in dom(f) \setminus \{\overrightarrow{x}\}$. Similarly, a function $f'' = f \ominus \overrightarrow{x}$ is such that $f''(\overrightarrow{x})$ is undefined and $f''(\overrightarrow{x'}) = f(\overrightarrow{x'})$ for all $\overrightarrow{x'} \in dom(f) \setminus \{\overrightarrow{x}\}$. The definitions of $\oplus$ and $\ominus$ can be extended to tuple sets, by iteratively applying the definition to all tuples in the set (noting that the order in which the elements are applied is not important).

When the system is in a given state S_i, the replaying of an event e causes the system to move to state S_j in which activity instance $(activity(e), case(e))$ is in state $state(e)$ and the process variables are updated according to the values associated with e.

Definition 5 (*System State Transition*) Let $S_i = (\alpha_i, \nu_i, \tau_i^s)$ be the current state during replay and e be the next event to

replay. Replaying e causes the current state S_i to change to state $S_{i+1} = (\alpha_{i+1}, \nu_{i+1}, \tau^s_{i+1})$. This change is denoted as $S_i \xrightarrow{e} S_{i+1}$, where

$\alpha_{i+1} = \alpha_i \oplus ((activity(e), case(e)), state(e))$
$\nu_{i+1} = \nu_i \oplus$
$\{((v, case(e)), properties(e, v)) \mid v \in dom(properties(e))\}$
if $state(e) = Executing$
$\tau^s_{i+1} = \tau^s_i \oplus ((activity(e), case(e)), timestamp(e))$
else if $state(e) = Concluded$
$\tau^s_{i+1} = \tau^s_i \ominus (activity(e), case(e))$
else $\tau^s_{i+1} = \tau^s_i$

The initial state from which replaying starts is $S_0 = (\alpha_0, \nu_0, \tau^s_0)$ where $dom(\alpha_0) = dom(\nu_0) = dom(\tau^s_0) = \varnothing$. Replaying is used to reconstruct the *execution history*.

Definition 6 (*Execution History*) Let $\langle e_1, \ldots, e_n \rangle$ be the sequence of events recorded in an event log, i.e. for every $1 \leq i < j \leq n$, $timestamp(e_j) \geq timestamp(e_i)$. Let $S_0 \xrightarrow{e_1} S_1 \xrightarrow{e_2} \cdots \xrightarrow{e_n} S_n$ be the sequence of states visited when replaying the event log. An *execution history* is a sequence of pairs $H = \langle (S_1, t_1), \ldots, (S_n, t_n) \rangle$ where (S_i, t_i) denotes that the system entered state S_i at time $t_i = timestamp(e_i)$.

2.3 Projecting states onto maps

Activity instances are visualised as pie charts on a map. By not fixing the type of map, but by allowing this choice to be configurable, different types of relationships can be shown on different maps, thus facilitating the extraction of deeper insights into processes in various contexts. Many types of maps can be thought of: geographical maps (e.g. the map of a university's campus), process schemas, organisational diagrams, Gantt charts, and others. Naturally, one can also make highly specialised maps to suit a particular purpose.

Let M be the set of maps of interest, each identified by their name. In addition to variables in V, we introduce special variables $\mathcal{T}$ and t to incorporate references to time. The former represents the starting time of an activity instance and the latter the current time during a log replay. The purposes of these two special variables will become clear later in this section.

The positioning of an activity instance may vary across different maps. When the use of a certain map is envisaged, the location of activity instances at run-time on this map should be captured through a formal expression specified at design time by process analysts. This is done by defining a position function for each map $m \in M$.

Definition 7 (*Position function*) A position function for a map $m \in M$ is a partial function that returns a pair of expressions for each activity type

$$position_m : A \nrightarrow Expr(V \cup \{\mathcal{T}, t\}) \times Expr(V \cup \{\mathcal{T}, t\})$$

where, given a set X of variables, $Expr(X)$ is the domain of all logical expressions that use some of the symbols in X.[5]

For each activity instance $ai = (at, cid)$ and each map m, $position_m$ returns a pair of expressions. The evaluation of these expressions, over a state S, returns a pair of coordinates (x, y), which is the position of ai on map m in state S.

Function $position_m$ may also be partial as some activity types are not naturally associated with a position on map m. This may simply be because it is not relevant or it is meaningless to project those instances onto a particular map. For example, it is meaningless to project automated system tasks onto a timeline map as these tasks tend to finish very quickly, hence, they do not add much value to a timeline map visualisation.

Let $\xi : X \nrightarrow U$ be a value assignment of a subset of the variable names in X. We define *eval* as a function which, given an expression $e \in Expr(X)$ and a value assignment ξ, yields an integer number obtained by evaluation e on ξ: $eval[\![f]\!]\xi = c \in \mathbb{Z}$. The evaluation may also be not possible: e is defined over a variable $x \in X$, which does not take on a value according to ξ (i.e. $x \notin dom(\xi)$). In this case, $eval[\![e]\!]\xi = \perp$.

Definition 8 (*Projection of an activity instance on a map*) Let us consider a map $m \in M$ associated with a position function $position_m$. Let us also consider a state $S_i = (\alpha_i, \nu_i, \tau^s_i)$ and an activity instance $ai = (at, cid) \in dom(\alpha_i)$ such that $at \in dom(position_m)$. Denoted $position_m(at) = (e^x_{m,at}, e^y_{m,at})$, the coordinates to which to project a pie chart for ai on map m for state S_i at a given timestamp $\bar{t}$ are

$$coord_m(ai)\big|_{p_m, S_i, \bar{t}} = \left(eval[\![e^x_{m,at}\xi_{ai}]\!], eval[\![e^y_{m,at}\xi_{ai}]\!]\right)$$

where $\xi_{ai}(\mathcal{T}) = \tau^s_i(ai)$, $\xi_{ai}(t) = \bar{t}$, and for all $vn \in V$, $\xi_{ai}(vn) = \nu_i(vn, cid)$.

Conceptually, the projection of an activity instance will result in one pie chart that is made up of precisely one slice, unless (1) the position function is not defined for the corresponding activity type or (2) the projection coordinates take on a negative or indefinite (i.e. $\perp$) value for x or y. Later, in Sect. 2.4, we discuss how multiple one-slice pie charts that are projected onto exactly the same position can be merged to form a pie chart with multiple slices. To concretely illustrate how position functions can be defined, we introduce two examples.

[5] We intentionally do not elaborate more on the type of expressions supported. The expressiveness is bounded by the actual technology that is employed. As discussed in Sect. 3, our implementation supports the entire range of expressions that can be defined through the XQuery language.

Example 1 This example refers to the map of Australia (denoted as *au*) of which a state projection is shown in Fig. 1. Activity instances that are active in each state are projected according to the following position function:

$$position_{au}(at) = \begin{cases} (743, 530) & \text{if } area = \texttt{NSW} \\ (697, 633) & \text{if } area = \texttt{VIC} \\ (489, 491) & \text{if } area = \texttt{SA} \\ (213, 374) & \text{if } area = \texttt{WA} \\ (758, 739) & \text{if } area = \texttt{TAS} \\ (470, 247) & \text{if } area = \texttt{NT} \\ (724, 334) & \text{if } area = \texttt{QLD} \end{cases}$$

The variable area (which exists in the log) encodes the geographical region of the claimants. These pairs of coordinates correspond to pixels in the map of Australia that are located within the borders of the corresponding Australian region. For instance, the coordinate pair (743, 530) corresponds to a pixel inside the border of the region of New South Wales.

Of course, this is a very simple and straightforward position function. In practice, one may need to define more complex position functions that involve multiple variables, such as shown in the second example below.

Example 2 Let us consider a loan request process where each process instance corresponds to a different request. The applicants' monthly income and the requested loan amount are stored in variables *income* and *loan*. One can define a cartesian map c where every activity instance is associated with a distinct pie chart whose x and y-coordinates are determined by the values of these variables. Assume that the maximum values of *income* and *loan*, as seen in the log, are 150,000 and 10,000, respectively. Also, assume that the maximum $x-$ and y-coordinate values on to which a pie chart can still be properly displayed on map c are 800 and 600, respectively. To ensure that an activity instance with maximum *income* and *loan* values can be properly displayed, we can define a position function such that $position_c(at) = (income \cdot \frac{800}{150{,}000}, loan \cdot \frac{600}{10{,}000})$.

So far, we have defined how to project activity instances onto a map. Formally, a *photograph* is composed of three sets, which, respectively, include pie charts with valid, invalid, and no coordinates. A *movie* is a sequence of photographs.

Definition 9 (*Photographs and Movies for a map*) Let $H = \langle (S_1, t_1), \ldots, (S_n, t_n) \rangle$ be an execution history. Let m be a map associated with a position function $position_m$ and m be a timestamp between t_1 and t_n. Let $S_i = (\alpha_i, \nu_i, \tau_i^s)$ be the state of the system at time t, i.e. $t_i \leq t < t_{i+1}$. Let us define an activity instance (at, cid) to be active in S_i if $(at, cid) \in dom(\alpha_i)$ and $\alpha_i(at, cid) \neq Concluded$. The *photograph* for map m at time t is a triple (P, N, I) consisting of a set $P \in (A \times V \in \mathbb{N}_0 \times \mathbb{N}_0)$ of pie charts with valid coordinates, a set $N \in (A \times V)$ of pie charts with no coordinates and a set $I \in (A \times V)$ of pie charts with invalid coordinates. For each active activity instance $ai = (at, cid)$, if $at \notin dom(position_m)$, then $(at, cid) \in N$. Otherwise, denoting $(x, y) = coord_m(ai)\big|_{pos, S_i, t}$, if $x \neq \perp$ and $y \neq \perp$ and (x, y) is in the boundaries of the image for map m, then $(at, cid, x, y) \in P$, else $(at, cid) \in I$.

In our framework, the coordinates $(0, 0)$ are located at the top-left corner of a map with the x and y axes directed towards the right and the bottom, respectively, as their values increase. Therefore, pie charts will be displayed on a map if (1) $x > 0$ and $y > 0$ and (2) if the values of x and/or y-coordinates are not too large that they are projected outside the boundary of the underlying map. If x or y is negative, the coordinates are considered as *invalid*. Conversely, the activity instances are considered to have *no associated position* if the position function is undefined for the respective activity types or the evaluation of the position is not possible. For a photograph (P, N, I), the elements in P represent pie charts that can be projected onto maps. The activity instances in N with no associate position are enumerated in a separate list, as are the activity instances associated with invalid positions.

Loops and Multiple Activity Instances. As per Definition 1, activity instances are univocally identified by a pair consisting of an activity type and a case identifier. This definition may pose an issue in distinguishing two activity instances of the same type that are executed within the same case (as, by definition, these two activity instances will be the same). This situation can happen when there are (1) loops in processes, and/or (2) concurrent executions of multiple activity instances of the same activity type within the same case (henceforth, referred to as *multiple activity instances*).

Logically, when two activity instances of the same type are sequentially being executed within the same case, for the purpose of replaying event logs on maps, this does not pose problems, since the sequence of states (built from the execution history of the event log) will capture these two activity instances at different states. For example, assume that an activity type (at_x) is executed twice within a particular case. The first execution of this activity instance (c_x, at_x) may be captured within, let us say, the first three states of the execution history (S_1, S_2, S_3). From S_4 onwards, $\alpha(c_x, at_x)$ is *completed* and will not be projected onto the map. Assume that the same activity type at_x becomes active at the seventh state of the execution history (S_7). At S_7, the state of the "same" activity instance (c_x, at_x) will change again into, for example, "scheduled"; hence, it will be projected onto the map again. Therefore, we can see that our framework still projects activity instances as they should be, even if there is a loop. For readers who would like to see a more elaborate proof of the ability of our framework to handle loop, we refer to "Appendix 8".

Our visualisation framework is able to project multiple activity instances running in *different cases* (e.g. multiple concurrent payments by different users when buying different goods in an online retail system). However, our framework cannot properly project *multiple activity instances* that are concurrently executed *within the same case* (e.g. multiple concurrent payments by the same user when buying some goods in an online retail system). This limitation is due to the fact that these concurrently executed activity instances (within the same case) would be treated as events referring to one and the same activity instance and, hence, would only result in one pie chart being projected.

Our framework could easily be extended to cope with this problem (e.g. by adding an activity instance identifier to each activity instance). However, our experience shows that, often, event logs do not contain enough information to differentiate those instances. Moreover, this pattern of multiple concurrent activity instances of the same type within the same case is almost never present. Therefore, we believe that it is not worthwhile to overcomplicate the framework (and the implementation) to address this problem.

2.4 Merging overlapping pie charts

Different activity instances can be positioned at the same coordinates and thus be overlapping. In these cases, our framework (and the implementation) merges those pie charts to form large pie charts in order to avoid the situation where one pie chart hides other pie charts centred at the same coordinates. The diameter of such pie charts grows according to the number of activity instances involved. This means that given two execution history states s_1 and s_2, if the number of activity instances (n) that are projected on the same coordinates (x, y) is higher in s_1 than s_2, then the diameter of the pie chart centred at (x, y) is larger when projecting s_1 than when projecting s_2. For instance, if all instances of an activity type at are projected onto certain coordinates, a larger pie chart at state s_1, compared to s_2, implies that more instance of at were active in state s_1 than in state s_2.

Li et al. [20] conducted an analysis with a number of subjects where they found that a quantity n represented as circles is most intuitively perceived when the circle grows to the power of 0.4: $n^{0.4}$. We apply this observation to our case where n pie charts of different activity instances are joined. The sizes of the slices in the merged pie chart are determined by the chosen colouring scheme (see Sect. 2.5).

A different situation is when pie charts partially overlap, which means that the pie-chart centres do not coincide but the areas intersect. Alternatively, it is also possible that one larger pie chart "hides" another smaller pie chart due (even though the centres of these pie charts do not coincide). In order to deal with this case, all pie charts are filled with colours with a certain degree of transparency. In this way, one can easily see the areas, the slices, and the sizes of the pie charts.

The drawback is that the overlapping areas may be coloured differently and these colours are partly transparent. As a result, the colour may differ from those with which the areas are filled, leading to confusion (due to the unintended generation of "new colour" based on the combination of the various overlapping semi-transparent colours). The framework also allows partially overlapping pie charts to be merged, as if they centred at the same coordinates. The natural solution is to centre the resulting pie chart at coordinates that are the geometric averages of the coordinates of the pie charts involved. The experience shows that merging partly overlapping pie charts is usually not preferable since too many pie charts tend to be merged and, hence, positioned at coordinates that have little to do with the coordinates of the original pie charts.

2.5 Colouring schemas for pie charts and fading

The slices of the pie charts (or the entire pie chart when not obtained by merging) can be coloured according to some given characteristics of the associated activity instances. Currently, three colouring schemes are supported and they are based on (1) the state of the activity instances, (2) the characteristics of the activity instances, or (3) the age of the activity instances.

Our framework can be extended to allow users to write their own "colour functions" to determine how pie charts should be coloured. However, our current implementation does not support such an extension yet. We are not sure whether such an extension will be useful as the existing implementation already requires users to make substantial decisions for it to be usable. Adding more options may in fact reduce usability and raise the issue of choice paralysis.

- *On the state of the activity instance.* When this scheme is used, an activity instance ai is coloured according to $\alpha_i(ai)$ where $S_i = (\alpha_i, \nu_i, \tau_i^s)$ is the current state. In particular, for activity instances that are in the scheduled, allocated, executing, or suspended state, we fill the relative pie chart with white, cyan, green, and black, respectively. Of course, activity instances that are concluded or have not been scheduled yet are not represented on a map.
- *On the characteristics of activity instances.* When this scheme is used, the end-user chooses one of the variables $vn \in V$ present in the event log. The value of the selected variable determines which colour is used to fill the pie charts of the activity instances. Let $S_i = (\alpha_i, \nu_i, \tau_i^s)$ be the current state. A slice in the pie chart corresponding to an activity instance $ai = (at, cid) \in dom(\alpha_i)$ is coloured according to $\nu_n(vn, cid)$. In particular, the 15 most commonly occurring variable values are associated with the

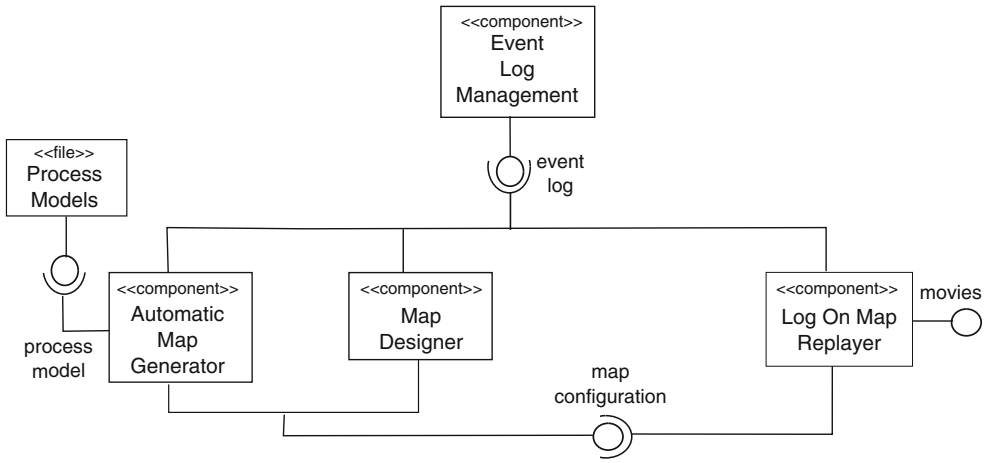


Fig. 3 Architecture of the implementation

15 non-white colours of the 16-colour EGA palette.[6] The white colour is excluded as it is used to represent all other variable values. Similar to the previous colouring scheme, the colours in the pie charts will be updated as soon as the value of the chosen variable $vn \in V$ changes as per the *execution history*.

– *On the age of activity instance.* In this approach, pie charts are coloured according to the age of an activity instance, i.e. the amount of time that has elapsed since the instance was started. The colour white is associated with pie charts (or slices) for activity instances that just started (very recent). As time progresses, the colour becomes closer and closer to black. Let $S_i = (\alpha_i, v_i, \tau_i^s)$ be the state at time t, then the recency of an activity instance $ai = (at, cid) \in dom(\alpha_i)$ is computed as follows: $recency(ai) = \exp^{-\frac{\log_e(2)(t-\tau_i^s(ai))}{MET(at)}}$ where $MET(at)$ is the average of the time that was taken to complete instances of activity type at. For each activity instance ai, $recency(ai)$ is always between 0 and 1. If $t = \tau_i^s(ai)$, then it means that activity instance ai was just started, hence $recency(ai) = 1$. Value $recency(ai)$ decreases exponentially as ai ages. When $t - \tau_i^s(ai) = MET(at)$, $recency(ai) = 0.5$. An established approach is used to map $recency(ai)$ to a colour: the "Fire Colour Palette" [29]. The colour ranges in intensity from a bright white ($recency(ai) = 1$) through yellow, orange ($recency(ai) \sim 0.5$), brown, and then to black (as $recency(ai) \approx 0$).

3 Implementation of the framework

Figure 3 shows the architecture of the implementation. The *Map Designer* component is implemented as a stand-alone Java application, whereas components *Automatic Map Generator* and *Log On Map Replayer* are implemented as a plug-in of ProM, an open-source "pluggable" framework for the implementation of process mining tools in a standardised environment.[7] The *Event-Log Management* component is an existing component of ProM, responsible for importing and filtering event logs. ProM supports the import of many different file formats for event logs and it natively supports the Extensible Event Stream (XES) format (cf. Sect. 2.1). A ProM plug-in requires a number of input objects and produces one or more output objects. These input objects could, for example, be event logs or output objects of other plug-ins. In this way, one can define a chain of invocations of plug-ins.

3.1 Replaying logs on maps

The *Log On Map Replayer* is the core component of our visualisation framework. It implements the approach defined in Sect. 2. It takes a map-specification file and an event log as input. The map-specification file is in XML format. It consists of a set of available maps (i.e. the map name and the URL where the map image can be retrieved) and the corresponding position functions for each map.

Starting from the event log, the sequence of system state is built. Each system state is represented as an XML document containing the information as per Definition 4. A position function is realised as a pair of XQuery expressions over such a document. Consistent with Definitions 7 and 9, the evaluation of these XQuery expressions yields the $x-$ and y-coordinates of the pie chart of each active activity instance in each photograph.[8]

[6] http://en.wikipedia.org/wiki/Enhanced_Graphics_Adapter.

[7] http://www.promtools.org.

[8] We use the XQuery processor SAXON, its library can be downloaded at http://saxon.sourceforge.net/.

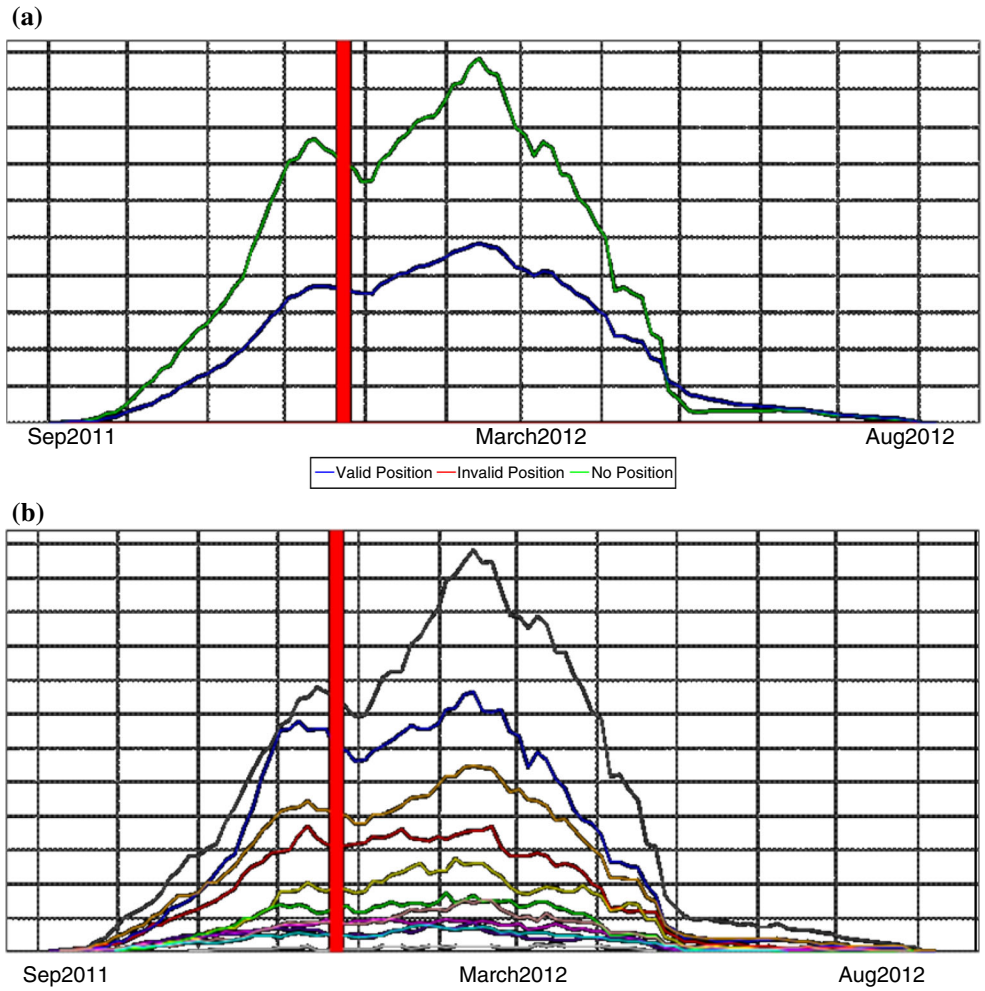


Fig. 4 Screenshots for the "position trend" graph and the "colour trend" graph as indicated on the graphical user interface are shown in Fig. 1, **a** A screenshot of the "position trend" graph, **b** A screenshot of the "colour trend" graph (colour figure online)

The set of movies produced, one per map, can be played using the graphical user interface provided by the *Log On Map Replayer* (note that movies have to be produced first before they can be played, i.e. they are not generated on the fly). Various types of interaction with the maps are supported by the component. Users can move back and forth, play the movies at different speeds, zoom in or out to aggregate or disaggregate pie charts. As the movie is being played, users can change the colour scheme to highlight certain characteristics of the activity instance, according to the schemes presented in Sect. 2.5. Furthermore, users can quickly jump to a photograph of interest by manipulating the slider bar that is provided by the tool.

We also provide two graphs to guide the user in selecting parts of the movie that may require a more detailed examination. Examples of these graphs are shown in Fig. 4. Figure 4a (top) shows the distribution of pie charts with valid positions, invalid positions, and no positions (cf. Sect. 2.3) at any given point in time, i.e. for each photograph. Figure 4b (bottom) shows the distribution of colours associated with pie charts that are visualised in each photograph. A user can click on a point of the graph, and the movie directly moves to that photograph. For instance, one may be interested to jump to a moment when there is a predominant number of activity instances with certain characteristics, which means that there is a large number of pie charts coloured in a certain way or projected at certain coordinates. Users can also move from one map to any other provided maps and compare the projection of the system state onto different maps, thus gaining an insight into the same system state from different perspectives.

3.2 Definition of position functions

Defining XQuery expressions is a task that requires technical expertise and should be carried out by business analysts who know to which aspects more attention should be paid and, hence, which maps that are worthy of consideration. If

business analysts do not possess the expertise, they should work in concert with IT personnel for the technical support.

A position function, i.e. a pair of XQuery expressions, can be defined in two ways. The first way is to use templates of maps that have a broader applicability and customise these templates for a specific purpose. This can be done through the *Automatic Map Generator* component, which allows generating maps with very little or, even, no effort. Of course, there are many specific maps that one may want to define and that cannot be obtained as per the above. Therefore, we provide a *Map Designer* component that supports users to define these maps by specifying the name and the image to use as background, as well as the pair of XQuery expressions (i.e. the position function).

The *Map Designer* has been implemented as a Java standalone application. It supports the design of the XQuery expressions which, otherwise, would be extremely tedious. It provides an editor to support the definition of XQuery expressions (e.g. by providing the list of variables that can be used in queries) and to verify their correctness. Moreover, if an example event log is given as input, a preview of one of the photographs is provided for each map, thus allowing users to verify whether the projection functions that have been defined will produce the expected outputs. There may be situations in which every active instance of some given activity types is always positioned at certain coordinates regardless of the assignment of values to case variables (i.e. the value's assignment to variables as encode in function ν). The drag-and-drop feature is very helpful if one wants to define a map where activity instances are projected onto geographical maps, for example a building map to show the rooms in which various activity instances are carried out. Section 4 illustrates the interface of the *Map Designer* application and the *Log On Map Replayer* plug-in using our case study.

The *Automatic Map Generator* has been implemented as a ProM plug-in. It takes an event log as input and optionally a process model and produces a map, which consists of a name and a background image, and the corresponding position function. This component supports the generation of three types of map. We opted for these types after analysing the maps that are valuable for two use cases discussed in Sect. 4. In particular, we generalised these map types by introducing a number of customisable parameters. The number and complexity of these parameters have been maintained as low as possible so as to allow for the automatic generation of a sufficiently large variety of maps (and position functions), while keeping this step simple enough.

Some common parameters of all map types are their width w and height h. The three following map types are supported; they are as follows:

Cartesian Map. This type of map takes inspiration from the Cartesian coordinate system. The projection of activity instances is determined by choosing two numerical variables, v_x and v_y, from which the values of the x and y-coordinates are derived. Let $x_{\max}$ ($x_{\min}$) and $y_{\max}$ ($y_{\min}$) be the maximum (minimum) values for v_x and v_y as present in the event log. The position function for a Cartesian map c for instances of an activity type $at \in A$ is defined as $position_c(at) = (x'_{at}, y'_{at})$ where

$$x'_{at} = \frac{v_x - x_{\min}}{x_{\max} - x_{\min}} \cdot w \text{ and } y'_{at} = \frac{v_y - y_{\min}}{y_{\max} - y_{\min}} \cdot h.$$

For example, suppose that the end-user chooses variables *payout* and *amount* as v_x and v_y. Based on these variables, the plug-in determines the corresponding minimum and maximum values as seen in the log, e.g. $x_{\min} = 0, x_{\max} = 100, y_{\min} = 1000, y_{\max} = 10,000$. Opting for a map of size 800×600 pixels ($w = 800$ and $h = 600$), the plug-in thus specifies $x'_{at} = \big((payout - 0)/(100 - 0)\big) \cdot 800$ and $y'_{at} = \big((amount - 100)/(10,000 - 1000)\big) \cdot 600$.

Deadline/Timeline Map. Activity instances are positioned along the x-axis according to the time that they have been active. When an activity instance has just become active, its x-coordinate is equal to w. Its y-coordinate is obtained by choosing a numerical variable v_y, extracting its current value, and using the same computation as for the Cartesian map. As time progresses, the x-coordinate changes (becoming smaller and smaller), but the y-coordinate remains constant.[9] The position function for a time map l for instances of an activity type $ai = (at, cid)$ is defined as $position_l(at) = (x''_{at}, y''_{at})$ where

$$x''_{at} = \left(1 - \frac{t - \mathcal{T}}{d_{at}}\right) \cdot w \text{ and } y''_{at} = \frac{v_y - y_{\min}}{y_{\max} - y_{\min}} \cdot h$$

where d_{at} is a constant which defines the maximum valid duration for instances of an activity type at. As discussed in Definition 7, variable $\mathcal{T}$ identifies the timestamp where ai started. If a certain activity instance is active for more than d_{at}, x''_{at} will become negative and is then enumerated in the list of activity instances with invalid coordinates.

Process Model Map. As mentioned before, for a process model map, activity instances are projected onto the corresponding icon in the model. In order to enable the automatic generation of maps of this type, end-users need to provide a process model which also encodes the actual coordinates of the icons that represent the activity types defined in the model. Currently, we only support the Petri net formalism, though we believe that it is relatively easy

[9] Originally, the x-coordinate increased as time progressed; however, as discussed in Sect. 5.2, this created confusion with some of our evaluation subjects. Therefore, in our current implementation, we switched the direction of the movement of the pie charts.

to extend the implementation to support other process modelling formalisms. Petri nets are stored in files in PNML standard format,[10] which also encodes the coordinates of positions of all activities (i.e. Petri net transitions) of the model.[11]

4 A case study with an Australian insurance company

We applied the visualisation framework for one of the processes that are executed by Suncorp, a large insurance company in Australia. Through regular meetings (almost weekly) with the stakeholders from Suncorp, we identified the need to communicate the current landscape of Suncorp's claims processing performance to higher-level managers within the company. To this end, we believed that the visualisation framework proposed in this article can be used to generate a number of movies summarising Suncorp's claims processing trends and performance.

Suncorp provided us with data related to the processing of claims that were finalised in second half of 2012. It contains over one million events for 34 activity types, which together describe the processing of over 32,000 claims from multiple departments within Suncorp.

For the purpose of evaluating the visualisation framework and the usefulness of the resulting movies, a subset of the data, restricted to one department, was used. This subset of data was selected because it contains rich attribute information, including loss type (the cause of a loss that triggered an insurance claim, such as fire, theft, or burglary), payout amount (the amount of money paid to a customer as a result of an awarded insurance claim), team (the team within Suncorp which processed the claims), and many others. As will be detailed later, the richness of attribute information in this subset of the log allowed us to produce interesting maps and, consequently, movies.

These data contain 8,200 cases and 132,940 events. The reference implementation of our framework, that is, the *Log On Map Replayer*, was able to generate movies based on this subset of data. For the generation, we used a laptop equipped with a Intel Core i7 Processor at 2.20 GHz. The entire generation process took nearly 4 hours and used around 4 GB of memory. This illustrates that the framework and reference implementation can sufficiently scale to generate movies that show hundreds of thousands of events.

The reason for using a large amount of memory is related to the fact that each frame/photograph of each movie needs to be computed in advance and is kept in memory at all times. This is due to the fact that the computation of XQuery expressions is not sufficiently fast to be performed on the fly while each movie is being played. One new XQuery expression needs to be evaluated for each movie photograph and for each activity instance to be visualised (i.e. active at the time to which the photograph refers). The processing time required to generate the movie is not the major limitation as movies are generated offline. Moreover, these movies can be stored on disk and retrieved later for subsequent usage. For the Suncorp case, the movie files required nearly 2 GB. The issue is mostly related to memory; however, we designed the tool to use the available memory efficiently.

4.1 Overview of the maps (and movies) created for the case study

Using the data set described earlier, four movies were produced using four different maps. A thorough explanation of the interpretation of these maps and the various configuration options available to users (while a movie is being played) is provided in the remainder of this section. Some screencasts of the tool showing these four movies are available on our website.[12]

4.1.1 Map of Australia

The first movie was produced using a geographical map of Australia. In this movie, a pie chart is projected onto the map at the position that corresponds to the state or territory where the claim was based. Thus, the goal of this map is to display the distribution of insurance claims across all Australian states and territories at any given point in time, and how the distribution evolves over a period of time. Figure 1 shows a snapshot of the movie.

The bottom part of Fig. 1 shows the widgets to control the playback of the movie. From left to right, this area contains the play button to start/stop the movie; the timeline that shows the relative progression of the movie; and the clock showing the timestamp where the movie being played currently is and the slider to adjust the playback speed on the fly. It is important for the end-users to be able to adjust the playback speed. As also stated in [12], it is important to play the movie at the right speed. The optimal speed may be hard to predict: animations played too slowly may become boring, whereas the opposite can cause relevant information to be missed. The timeline shows a wave in which the x-axis represents time and the y-axis the number of active activity instances. It also contains a slider that represents the current point in the movie. The slider moves towards right as the movie is being played and be dragged by users to reach a particular photograph.

[10] http://www.pnml.org.

[11] In fact, the approach supports any type of process model notation; for instance, see Fig. 8. However, so far, only for Petri nets, we can automatically generate maps.

[12] http://www.processmining.org/online/logonmaps.

At the top of Fig. 1, we see the Maps panel that is made up of a number of tabs, each associated with a different movie. By selecting a tab, the corresponding movie is brought to the front to show the state of the process at a specific point in time, in terms of the activity instances that are active and their positions. While playing the animation, users can change the animation they are watching in case they wish to consider a different type of movie. Figure 1 shows a situation in which two animation movies are executed simultaneously: in foreground, the Australian map animation (called "Loss Cause/the State" on the tab), and in background, the so-called Quadrant Map (called "Incurred Amount/Claim Duration" on the tab and discussed in Sect. 4.1.2).

There are two boxes on the panel on the left-hand side of the screenshot. The top box enumerates the activity instances for which there are no associated positions (i.e. they do not belong to the domain of the corresponding position function). The bottom box enumerates those activity instances whose positions are "invalid" (i.e. the corresponding position function returns coordinates which either fall outside the map boundary or could not be evaluated). In the screenshot shown in Fig. 1, there was no activity instance that was projected as "invalid" at that particular point in time.

The centre of Fig. 1 shows a snapshot of the movie being played. The configuration file used to produce this movie projects activity instances onto the corresponding Australian state/territory in which the claim was lodged. The size of the pie charts is positively correlated with the number of events that are simultaneously present on the map at exactly the same position. As mentioned previously, pie charts projected at the same coordinates are merged to form large pie charts; moreover, the colouring scheme of the pie charts is configurable. In Fig. 1, each pie chart is configured to be coloured according to the "loss cause" of the insurance claims. The association of colours with values of loss cause is explained in the legend shown on the right-hand side panel.

As mentioned previously, each larger pie chart is divided in as many slices as the number of different colours associated with the merged pie charts. The size of the slice is proportional to the number of activity instances with a particular colour (such as a particular loss cause). The proportion of each slice (expressed as a percentage) can be viewed by rolling the mouse over the pie charts (as shown in Fig. 1).

On the top right-hand side of Fig. 1, there are two tabs (arranged vertically). The top one is labelled as "Parameters" and the bottom one is labelled as "Legend". As explained above, the purpose of the "Legend" tab was to describe the meaning of each colour in the pie chart. The "Parameters" tab, on the other hand, is used to configure the colouring scheme and the manner in which the pie charts are to be displayed in the animation movie.

Figure 5 shows the expanded parameter tab. The first checkbox allows a user to merge multiple pie charts into one pie chart even if the pie charts are only partially overlapping. If this option is ticked, pie charts whose areas partly overlap are merged. Otherwise, if it is unticked, pie charts are merged only if they are exactly positioned at the same coordinates. The second checkbox allows fading out pie charts when the corresponding activity instances are soon going to be concluded. The number of photographs required for the pie charts to begin to fade out can be customised using the slider on the panel. When pie charts are merged, by default, the bigger pie charts are annotated with the number of pie charts that are merged. Sometimes, process owner does not want to disclose this aggregate information for privacy and/or confidentiality reasons. For instance, this happened for the Suncorp's case study. Therefore, the last checkbox allows users to display or remove the exact number of merged pie charts that is shown in the middle of each pie chart: If this option is unticked, the number will not be displayed in the movie (the legend will also show an "N.A." status in place of the exact number).

The second half of the parameter panel allows users to customise the colouring scheme of the pie charts in the movie according to one of the schemes described in Sect. 2. The three options will enable the colouring of pie charts based on the state, the age, or the values of a particular variable of the corresponding activity instances, respectively. Furthermore, for the third option, the variable type to be used is selected from a drop-down box, as shown in Fig. 5.

Figure 4a, which has been previously discussed, refers to the evolution of the number of activity instances that are projected onto this map of Australia and the colours of the associated pie charts. These graphs can allow one to gain interesting insights. For this map, it provides a helicopter view of the distribution and the characteristics of claims across Australian states and, more importantly, the differences between these characteristics. For example, while playing the movie, we noticed that the state of Tasmania had a higher proportion of claims from damages of rental properties when compared to other states, while natural hazards seem to be one of the most dominant causes for insurance claims across all states.

Based on the description of our framework in Sect. 2, the *Log On Map Replayer* plug-in is used to project individual activity instances onto the map. However, there are situations when stakeholders are more interested in the analysis of the overall distribution of cases and their characteristics, rather than of the various activity instances. This is precisely the situation that we encountered in our case study with Suncorp. In fact, two out of the four movies produced (notably the Australian map and the Quadrant map) are concerned with case-level performance.

For this purpose, the original event log was preprocessed. We introduced a new activity type named "Case". In this way, we ensured that a different instance of this activity existed for each case, which started at the same time as the first actual activity instance and completed at the same time as the last

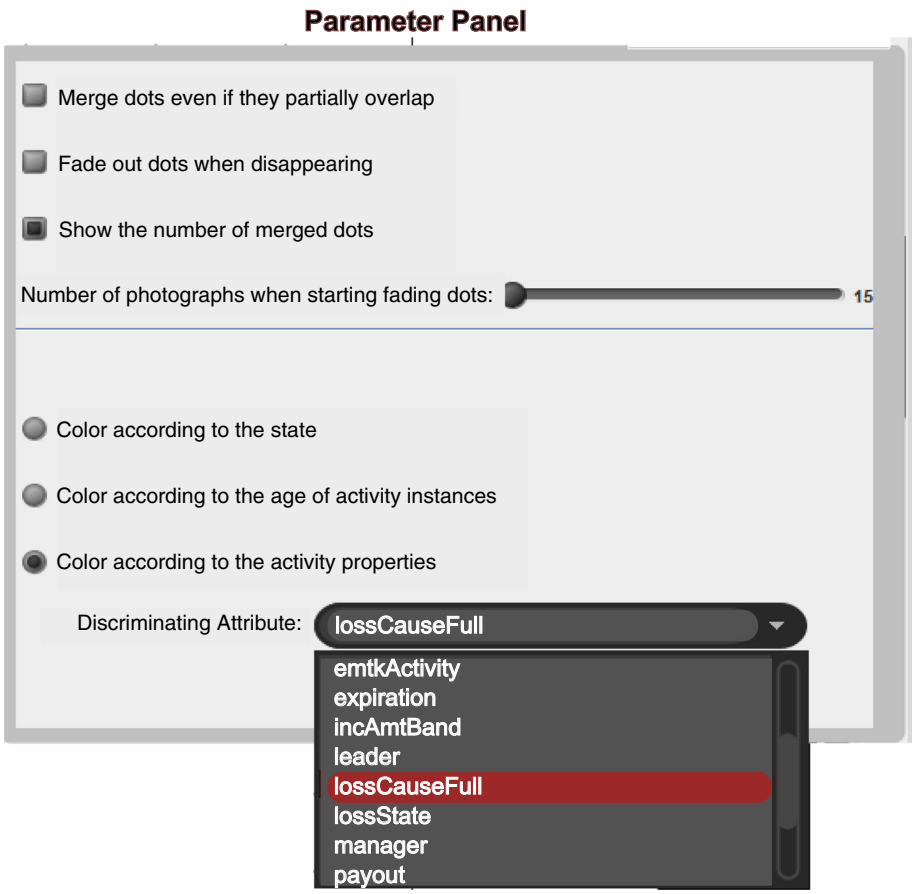


Fig. 5 A screenshot of the interactive parameter panel

actual activity instance. The event log was enriched accordingly. For the Australian and Quadrant map, we defined a projection function over activity type "Case" only. The result is that a new pie chart will appear (and possibly be merged with others) when a new case is started, and it will disappear when the case completed.

4.1.2 Quadrant map

The second movie was produced based on a Cartesian graph, whereby the x-axis represents the amount of insurance claim payouts, and the y-axis represents the number of days taken to process the claims. This map is called "quadrant map". Similar to the map of Australia, we only project the instances of the activity type "Case" on the map. These were added to the log as discussed in Sect. 4.1.1.

Figure 6 shows a screenshot of the generated movie where pie charts are coloured based on the age of the activity instances (see Sect. 2.3). Pie charts in the bottom-left quadrant represent claims with low payout values and relatively quick processing times (expected), pie charts in the top-right quadrant represent claims with high payout values and relatively long processing times (expected), pie charts in the bottom-right quadrant represent claims with high payout values and quick processing times, and finally, pie charts in the top-left quadrant represent claims with low payout values and long processing times (under-performing claims).

In other words, this movie allows us to gain insights into the performance of Suncorp's claims process over a period of time. In our case study, this movie proved to be useful in conveying to business analysts and managers the performance of claims processes.

4.1.3 Deadline map

The third movie was produced using a deadline map. It shows which activity instances were completed on time (before the deadline) and which activity instances were not (see Fig. 7). The position of a pie chart on the x-axis tells us the time remaining before the corresponding activity instance reaches its deadline, while the position of that pie chart on the y-axis reveals the type of that instance. In this figure, pie charts are coloured according to the teams that performed the corresponding activity instances.

When an activity instance becomes active, a pie chart representing the activity instance appears on the map. The x-axis position is initially determined by the amount of time the activity instance has before the deadline is reached (note

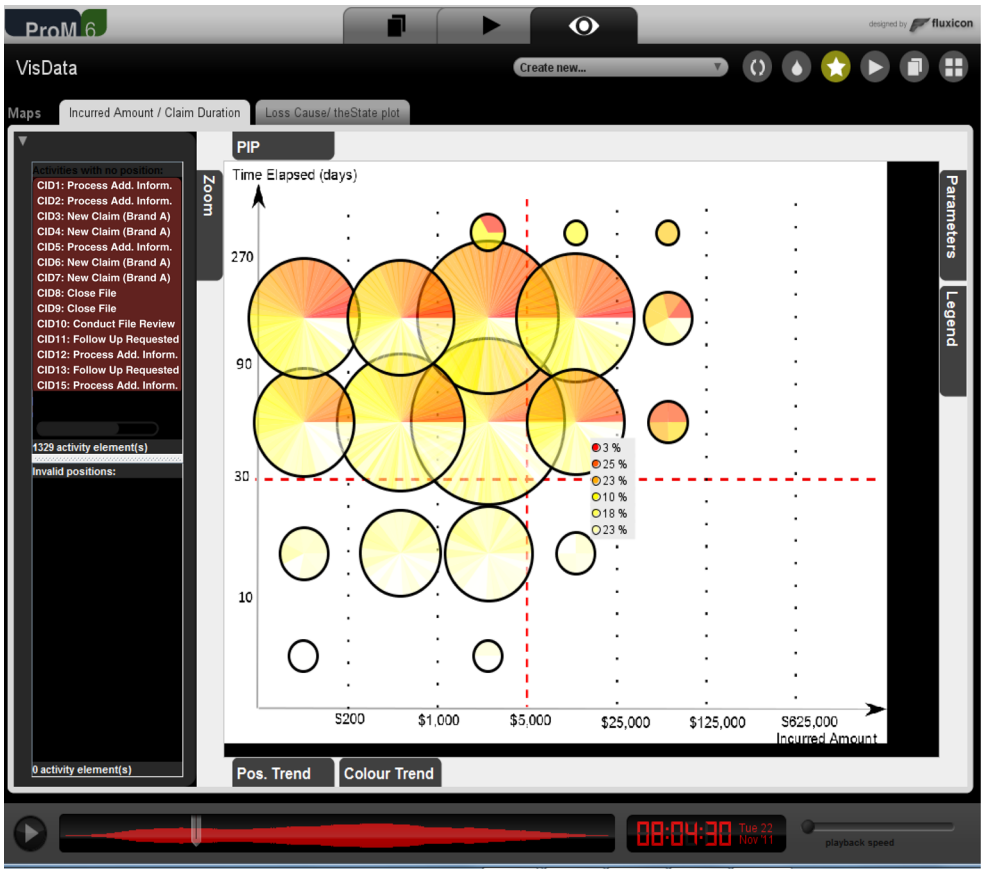


Fig. 6 A screenshot of the second animation produced using the quadrant map

that the deadline for every activity instance is provided as an event attribute in the log). The further the deadline of an activity instance is relative to the time when the instance becomes active, the further to the right the starting position of the pie chart is.

After an activity instance becomes active, the pie chart representing that activity instance moves from right to left as time progresses. The amount of time until the deadline expires is represented through the distance to the thick black line (i.e. the vertical line corresponding to 0 remaining days). We made the choice that this line does not correspond to the set of coordinate points with $x = 0$. We do so because we are still interested in visualising those activity instances whose deadlines have just recently expired.

In our deadline map, pie charts located at any coordinates with $x = 0$ means that the corresponding activity instances have expired about 12 days ago. Therefore, pie charts are associated with negative coordinates when the corresponding activity instance has expired more than 12 days ago. According to our framework, these instances are enumerated in the left-hand side list of activity instances with invalid coordinates.

In our case study, this movie allowed us to identify a number of activities that often ran overtime, such as the "Follow-Up Requested" activity, the "Conduct File Review" activity, and the "Incoming Correspondence" activity. Other activities, such as the "New Claim (IPI)" activity, mostly completed around the deadline.

4.1.4 Process model map

The fourth movie was produced based on a process model map (see Fig. 8). The process model used in this movie is the Fuzzy model [11] that we discovered using the Disco tool.[13] A Fuzzy model is a process model, whereby the nodes represent activities while the arcs represent the flows between activities. As explained in Sect. 3, pie charts are projected onto the map according to the position of the icon representing the activity captured by the pie charts. In this screenshot, pie charts are coloured according to the age of the activity instances.

A typical insight that can be gained from using this type of map is the identification of activities in a process that can potentially be a bottleneck. For example, while playing the movie, the appearance of a large pie chart on a particular activity icon over an extended period of

[13] http://www.fluxicon.com/disco.

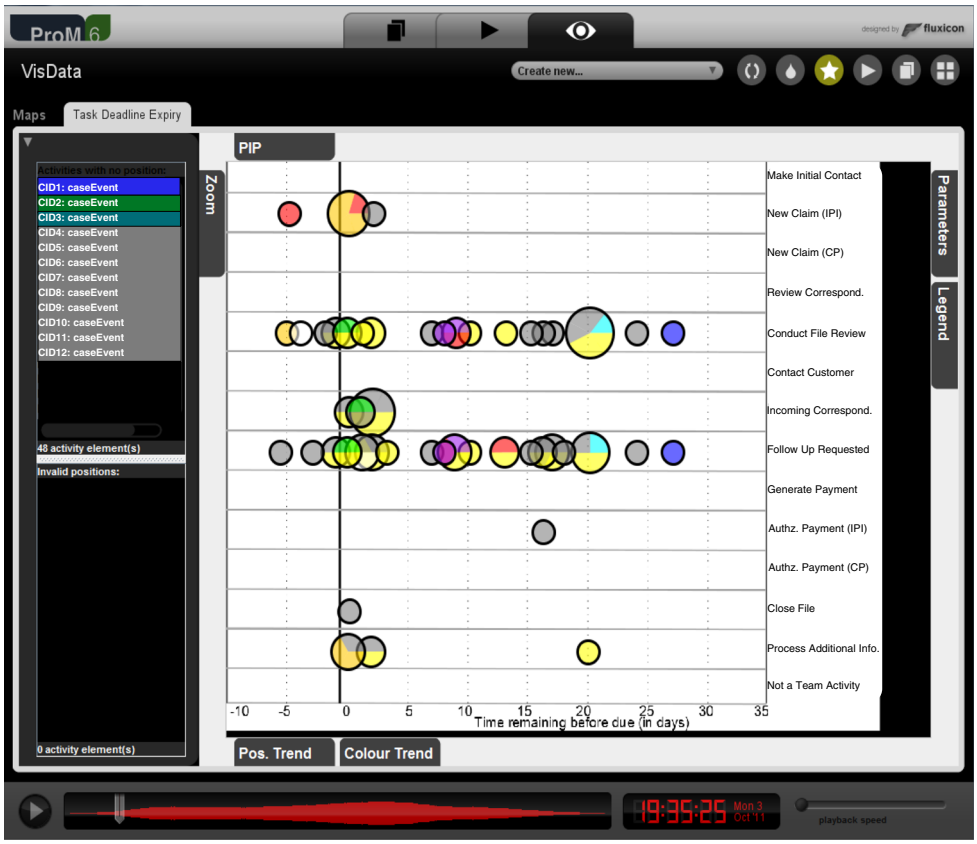


Fig. 7 A screenshot of the third animation movie produced using a deadline map

time may indicate the piling up of work items (or activity instances) related to that activity. As can be seen from Fig. 8, many activity instances were piling up for two activity types, namely "Follow-Up Requested" and "Conduct File Review".

4.2 Map designer

As stated in Sect. 3, a *Map Designer* tool has been implemented. Figure 9 shows two examples of how we have used the *Map Designer* to generate the configuration files for the four movies used in the case study.

Figure 9a shows how we have used the *Map Designer* tool to automatically generate the configuration file for the fourth movie (the process model movie). Here, we can see how a user can simply drag an activity name (from the "Task List" window) to the desired position on the map. By doing so, the *Map Designer* tool automatically generates a map configuration file which specifies, for each activity type whose instances need to be projected onto the map, the *static* positions of those instances.

Figure 9b shows how we used the *Map Designer* to help us generate the map configuration for the second movie (the quadrant movie). This movie requires a dynamic positioning of pie charts based on the variable values of the activity instances to be projected. Thus, to enable such a "dynamic" positioning of pie charts, we specified the desired XQuery statements in the corresponding pop-up window.

Overall, we found the *Map Designer* tool to be quite useful in enabling us to quickly define, and adapt, our map configuration files to suit the type of visualisation that we would like to see.

5 Evaluation with end-users

We performed an extensive evaluation to assess the validity, usefulness, and intuitiveness of the "process movie" metaphor. More specifically, we aimed to verify whether the metaphor of the map could be meaningfully understood by potential end-users and whether it allowed them to gain useful insights about their processes, which were unknown beforehand.

The evaluation was conducted in two rounds. In the second round, the evaluation was conducted using an improved version of the tool, which was obtained by addressing the issues encountered during the first evaluation round. This paper reports on the second version of this tool (which has also undergone even further improvements). Both rounds of eval-

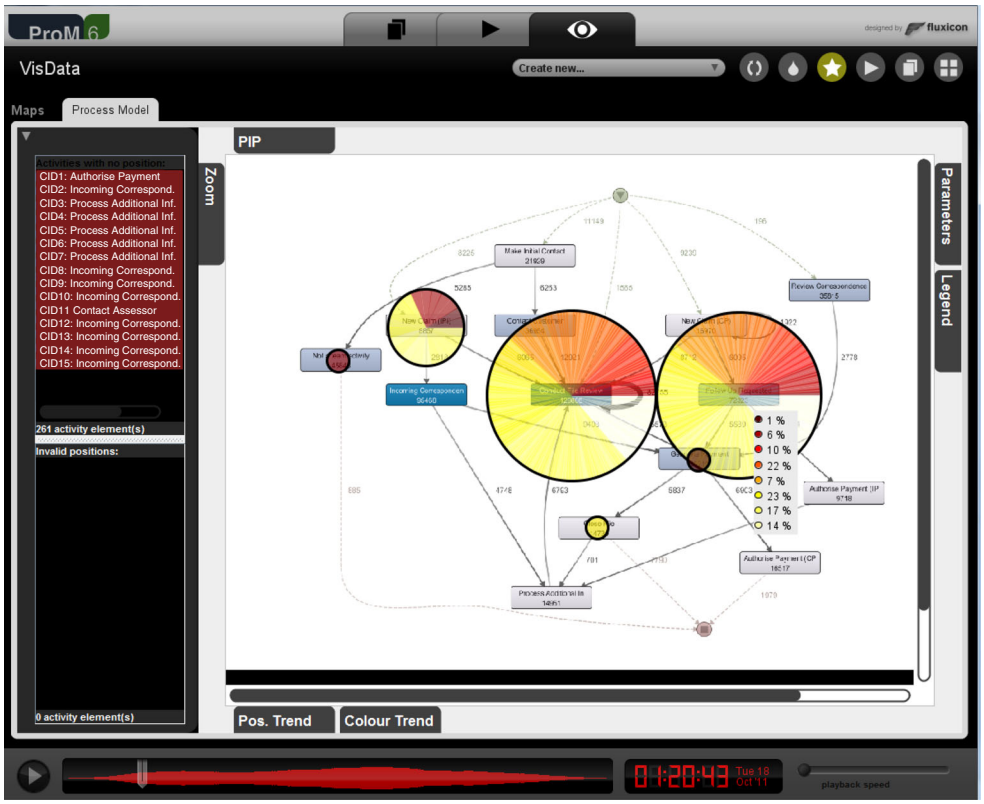


Fig. 8 A screenshot of the fourth animation movie produced using a process model map

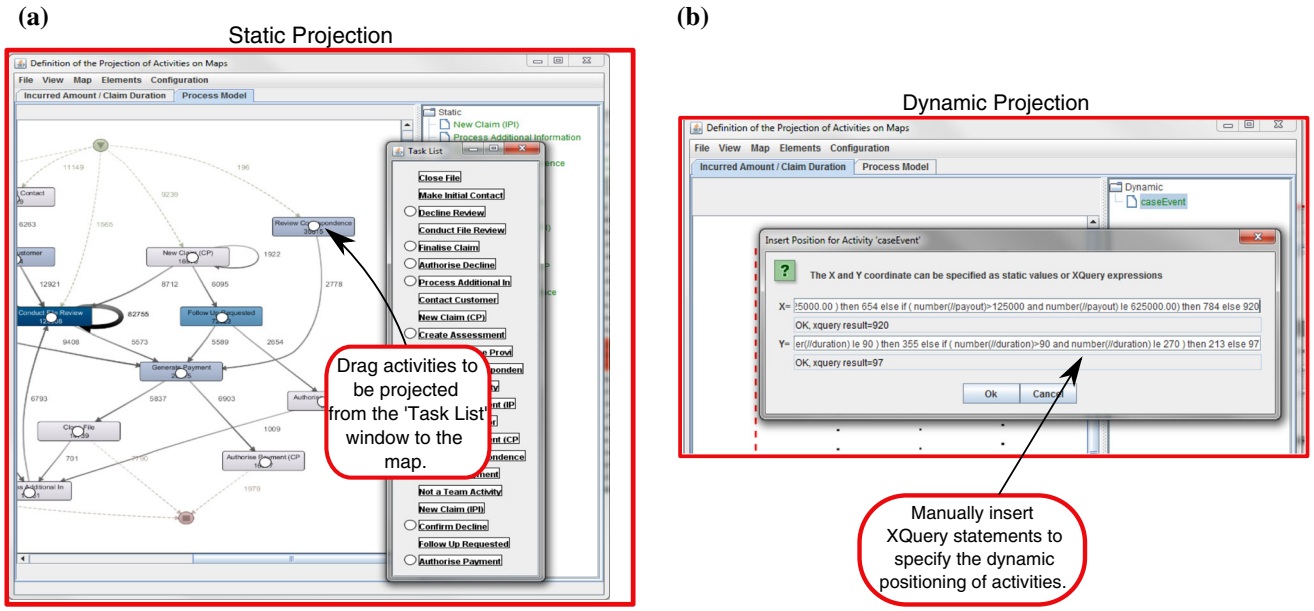


Fig. 9 Screenshots of the *Map Designer* showing static and dynamic activity projection. **a** Static activity projection, **b** dynamic activity projection

uations attempted to assess whether our framework, and the resulting tool, can indeed aid users in gaining useful insights into their processes.

We summarise the findings gathered during the first round in Sect. 5.1, whereas we elaborated on the second round of evaluations in Sect. 5.2.

5.1 First evaluation

In the first experiment, we used data related to the process for handling house-building permit applications submitted by residents of a Dutch municipality. We designed four movies based on four different maps. These maps include an organisational map that captures the different resource groups participating in the process, two process model maps, and a timeline map. The details of these four movies are available in [6].

Question. The question that we wanted to investigate was "Can users derive meaningful conclusion about the behaviours and/or performance of their processes?". We addressed this question by conducting an experiment involving the use of the tool by subjects from a Dutch municipality.

Methodology and Procedure. The methodology used was the *Thinking Out Loud* methodology [8]. The procedure of the experiment involved a brief explanation to the subject about the *Log On Map Replayer* tool to be used in the experiment, the four different movies produced, and the types of interaction with the movies that the tool allows. The users were then asked to observe the movies to draw meaningful conclusions about the process. While they were doing so, the participants were asked to verbalise their thoughts (such as insights gained by observing the movie). Such observations were written down for a later analysis.

Participants. This experiment involved three subjects working at the Dutch municipality. The participants had different levels of expertise with regard to analysing processes: one participant was a BPM specialist, another participant was a communication and marketing specialist, and the last one was a business advisor for customer contacts. Similarly, the degree of familiarity of the participants with the process being analysed also differed: the BPM specialist had no detailed knowledge about the process, while the other two participants were actively involved in the execution of the process.

Results. This evaluation round suggested that, by observing the movies and interacting with our tool, stakeholders managed to draw meaningful conclusions with regard to the behaviours and/or performance of their processes. Examples of conclusions that the participants drew include the accumulation of work items in a particular task of the process (while this was not the case in other tasks), the distribution of work items across various resource groups, and the phases in the process, whereby work items tend to accumulate the most.

The results of the experiment were encouraging confirming the benefits of our tool for stakeholders. Nonetheless, they also show various limitations of the initial version of our tool in terms of its usability and incompleteness. The major issues identified were (see [6] for details):

- in the first version of the tool, activity instances left no trace when they disappeared. The subjects found this particularly confusing as moving pie charts could disappear at different positions on the map.
- the interviewed subjects remarked that it was sometimes unclear how long activity instances were active.
- the characteristics of a case (e.g. the type of permit requested) could not be "made visible" in the movie (such as through colouring).

After addressing the issues above (and other minor issues), we released a second tool version, which is the version described in this article. To address the issues, the fading effect was introduced to make it more evident when activity instances are about to disappear (see end of Sect. 2). Furthermore, the different colour schemes discussed in Sect. 2.3 were introduced to relate activity instances to case characteristics or to draw attention to the recency of activity instances.

5.2 Second phase evaluation

After releasing the second version of the tool, we performed another round of evaluations with more participants. The participants of this second experiment session were nine Suncorp employees. The involvement of participants from a different continent allowed us to assess the framework in a different cultural context and work background. The movies used in the experiment were the four movies described in Sect. 4.

Question. The main question that we attempted to address with the second evaluation round was similar to the first: understanding whether, by using our tool, stakeholders can draw meaningful insights from their processes. In addition, the second experiment also sought to understand whether the usability issues identified in the first evaluation's round were actually solved.

Methodology and Procedure. The evaluation of the second version of our tool was conducted using the *Cooperative Evaluation* methodology, which is a mature, fully documented methodology in the field of human–computer interaction [8]. This is a cost-effective technique for identifying usability problems in prototype products and processes. This methodology is very similar to the *Thinking Out Loud* methodology except for the fact that the *Cooperative Evaluation* methodology also encourages both the design teams and users to collaborate in order to identify usability issues and their solutions. We found this flexibility to be useful as it allows us to probe deeper into what the participants were thinking (instead of being fully reliant on the participants' ability to convey what they were thinking accurately).

The experiment was conducted with each participant individually, one after another. Before the experiment started, we gave a brief introduction of the framework and of the four movies, after which we let the participant play with the tool on their own. Each participant was given 10 min to interact with the tool. In accordance with the methodology, no further comments were given, thus letting the participants draw their own conclusions. Without our interference, we could thus evaluate the level of understandability of the map metaphor and the usefulness of the approach when extracting knowledge from event logs.

While participants were interacting with the tool, they had to explain what they were doing by "thinking out loud". During the experiment, notes were taken on the behaviours exhibited by the participants (e.g. about difficulties in using the tool) and the conclusions that they managed to draw as a result of observing the animations and interacting with the tool. We used these notes later in the evaluation process to gauge the extent to which participants managed to draw meaningful insights from the movies.

At the end of the experiment, each subject was given the opportunity to fill out a semi-structured questionnaire with questions designed to measure the subject's impressions and expectations with regard to the tool. To ensure anonymity, the filled-out questionnaire was inserted into a ballot box by the participant him/herself, thus guaranteeing the anonymity of responses.

In our opinion, this methodology provides a valuable means to assess the validity, usefulness, and intuitiveness of the "process movie" methaphor, along with eliciting further possible improvement opportunities (see Sect. 5.3).

Participants. The participants of this experiment consisted of Suncorp employees with varying roles: five team leaders, one manager, two business analysts, and one claims' officer. The choice of participants was driven by trying to have subjects with different levels of knowledge of the insurance claim process, and different levels of familiarity with process mining techniques and business process management technology. As a matter of fact, three out of the nine participants were well aware of the existence of Business Process Management systems, while the rest were not aware at all, or only had a very limited awareness, of such systems. Furthermore, only one participant had good experience with process analysis, while the rest had no or limited experience.

Results obtained through the Cooperative Evaluation methodology. The application of the *Cooperative Evaluation* methodology allows participants to express their thoughts aloud, with particular focus on participants discussing interesting insights they gained, e.g. recurrent patterns. As explained earlier in this section, notes were taken regarding their high-voice remarks. All participants expressed the fact that they could discover patterns and learn insights that were not already known to him/her. For example, one participant gained insights with regard to the distribution of claim types across different Australian states and raised the possibility of how insurance premiums can be adjusted accordingly. A number of other participants gained insights into the "absurdity" of having small-value claims that had taken very long to complete. One participant clearly acknowledged the usefulness of the deadline map to compare performance levels across different teams. Another participant noted a peak time period in terms of the number of claims. Overall, the insights gained by participants from using this tool are consistent with what we expected the tool to be able to provide. The remainder of the observation notes taken during the experiment corroborated the results of the questionnaire, thus further validating the results of our experiment.

Questionnaire Results. As mentioned earlier in this section, the involved subjects were also asked to fill in a questionnaire. Figure 10 shows the eight questions of the questionnaire, along with a summary of the answers. Question Q3 was an open question where subjects could freely express what they wished or expected to find through the analysis. For each of the five remaining questions (i.e. Q4–Q8), the participants were expected to select one out of a number of predetermined satisfaction ratings (i.e. "very much", "much", "not so much", "not at all", and "I don't know"). Furthermore, each question has an empty section that participants could fill in to provide reasons for the satisfaction rating they selected.

The replies of participants to the questionnaire corroborated the observations obtained from the application of the *Cooperative Evaluation* methodology. The first two questions in the questionnaire (Q1 and Q2) were used to gauge participants familiarity with BPM systems and process analysis, whose results have already been summarised earlier. The rest of the questions (Q3–Q8) were used to gauge participants satisfaction with the tool.

We prepared a questionnaire with both closed and open questions. Through the third question (Q3), participants were asked about the type of process-related insights that they had expected to obtain by our tool. The types of answers varied but they were consistent with what one expect from a visual analytics tool. A baseline expectation from all participants was to understand the *trend* and *volume* of their claims processing in terms of performance, and to understand *why* certain trends occurred. Some participants expected the tool to guide them in identifying *problems* with their processes and how to resolve them. Finally, a few participants also expected the tool to help them identify *opportunities* in their processes with respect to planning the assignment of resources.

With respect to Q4, most participants (8 out of 9) found that the tool allowed them to gain process-related insights that

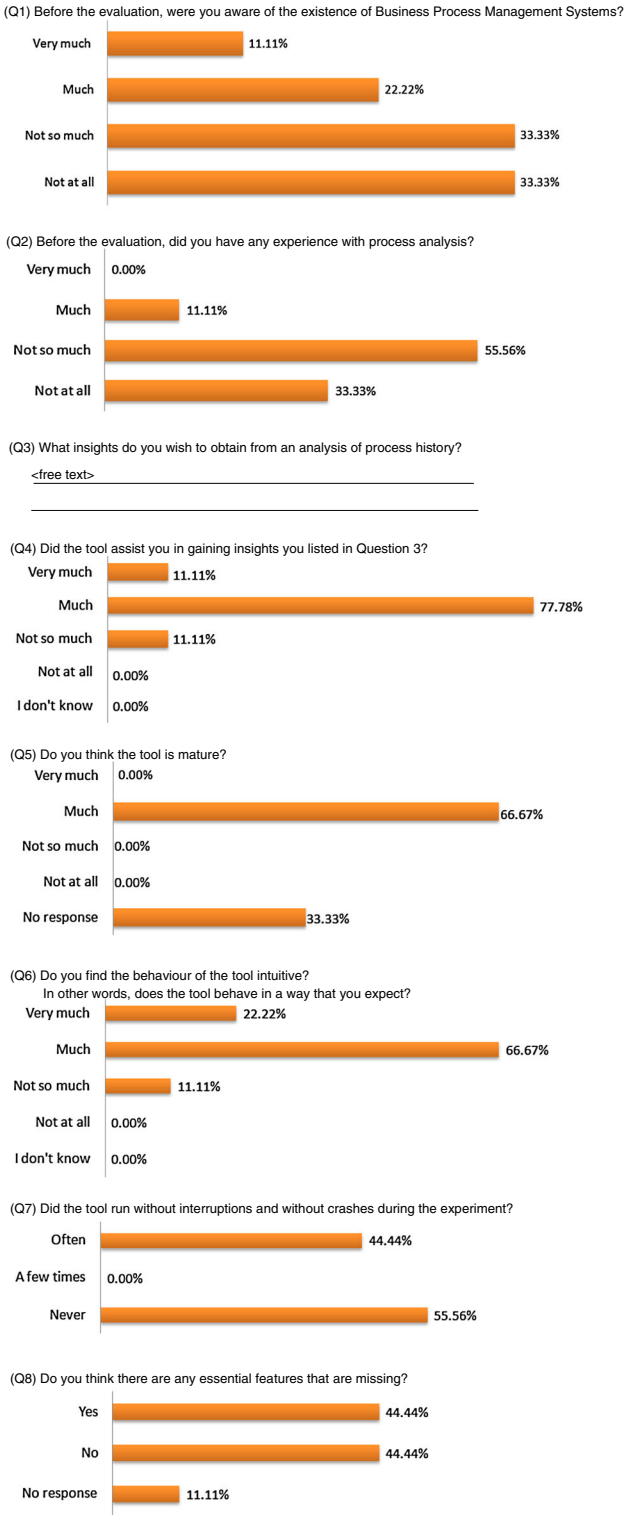


Fig. 10 Closed questions results. The "No Response" category is used to capture the situation where participants left the question unanswered (there was no such option in the questionnaire itself)

they expected (expressed as answer to Q3). This is confirmed by analysing the related comments, e.g. "*Having a visual representation makes it clearer* [. . .] *the flow on impacts when there is a problem in one area and how it then relates to another area.*", "*Was good to see key area for the business to improve on* [. . .]", "*You could clearly identify bottlenecks, claims with long duration but low value, etc.*", "*Give graphical representation of claims incidents and work loading. Give insights to claims costs compared to time of year*".

However, there was one participant who did not find the tool to help him/her much with addressing the question he/she listed earlier. Upon reading the comment, it turned out that the participant expected a feature in our tool which was simply not built: the ability to "drill down" into activity instances that appear on the movies to learn more about them (e.g. obtaining the details of the customer related to a particular activity instance). We understand that "drilling down" is a common feature of business intelligence tools and it is something that we can potentially include in our tool in the future (see Sect. 5.3 for details).

Another recurring comment was about the intuitiveness of the deadline map, with pie charts moving towards the left as time progresses. The participants expected them to move towards the right. Interestingly, in our first experiment, the pie charts did move towards the right; however, we changed the movement direction in the second version of the tool since the subjects from the Dutch municipality found the movements towards the right counter-intuitive. This makes us suspect that the intuitiveness of the pie charts movement direction is subjective and dependent on a user's cultural background.

In terms of the maturity of the tool (Q5), it was quite surprising to note that one-third of the participants did not provide any responses. However, by looking at the related comments, it turned out that these participants did not respond because they either did not understand the question or they found that they had not spent enough time with the tool to properly answer the question. Nevertheless, those who responded to this question all agreed that the tool is mature. The related comments also confirmed this observation, e.g. "*...it captures a lot of relevant information and provides plenty of options to navigate to specific areas you might want to examine...*", "*...it has all the activities for 6 months and I can go back in time if I wanted to. Obviously as we used it we would discover more things we may want to see but for the moment, I think there is enough for me to play with.*" At the same time, comments related to Q5 also suggested a number of possible tool improvements, including the use of "plain English" in the description and labelling of various configuration options.

The Q6 attempts to evaluate the intuitiveness of the tool. As shown in Fig. 10, the responses we obtained were quite similar, with eight out of nine participants finding the tool quite intuitive. From those who found the tool to be intuitive, we received comments such as "*It acts exactly to what you would expect, according to the selections you choose.*", "*Visual interpretation easy to accept.*", and "*[The] tool is*

easy to use." The one participant who did not find the tool to be intuitive commented on the need for the tool to have clear labels, filters, and "help" boxes so that "*we can easily navigate around it*."

Another feature of the tool that we wanted to evaluate was its operational stability. Therefore, Q7 asked participants to rate whether the tool ran smoothly during the experiment. The responses to this question are almost evenly divided, with five participants stating the tool ran smoothly during the experiment and the other four participants stating the opposite. One participant commented that it was frustrating that he/she managed to crash the tool, and other participants commented on the low-quality graphics used during the experiment due to the projector used.

Finally, we also asked participants whether there were any features in the tool that they would like to have but that were not currently available. Apart from one participant who did not respond, the responses to this question are evenly split, with four participants stating that there were missing features and the other four participants stating the opposite. Among those who responded "Yes" to this question, a number of features that should be added were suggested. These suggested features, except for those that have already been stated in response to Q6, will be detailed in Sect. 5.3.

5.3 Final remarks and research directions for future improvement

By analysing the responses to our questionnaire and the notes taken on the participants' interactions with the tool, we conclude that in most cases, the tool behaved as expected. However, there are various observations that deserve further attention (and which may be addressed as potential avenues for future work):

1. A number of participants highlighted that the language used in the tool was not written in "plain English", leading to a potentially wrong interpretation of the options or features supported by the tool.
2. The stability of the tool needs to be improved as it crashed rather frequently during the experiment.
3. A recurring request from many participants was to provide filtering capabilities so that a movie can be configured to display only information of interest (for example, to only display activity instances that were executed by a certain group of resources or that started and/or completed within a specific time period). It can be noted that ProM already has sophisticated log-filtering capabilities. Nevertheless, it may still be worthwhile to integrate them directly into this plug-in to allow users to do filtering while playing the movies.
4. One participant expressed his/her interest in being able to *drill down* into the activity instances associated with certain colours or having certain characteristics, and to extract the respective details.
5. For pie charts representing more than one activity instance, participants showed an interest in being able to quickly learn the percentage of slices of each represented colour (e.g. 30 % of slices are coloured red, 10 % blue, etc.). In the version evaluated, we simply showed the number of pie charts that were amalgamated, without detailing the statistics per colour.

The version reported on in this paper has already taken some of the points above into consideration. Language issues were solved. Regarding the second point, we performed more extensive software testing and solved the bugs that we encountered. Most of the bugs were related to the generation of the execution history, which, in turn, led to the production of wrong movies; some other bugs were related to the interaction of the end-users with the graphical interface. Point 5 has also been addressed: percentages are shown of the different colours used in pie-chart slices, instead of absolute numbers.

Another interesting direction for future work derives from the observation that pie charts have a large number of overlaps in practical case studies (both in the case of Suncorp and of the Dutch municipality). As alternative to a pie chart visualisation, we may visualise overlapping pie charts using bar charts. Here, we can build on the work by Simkin and Hastie [27].

5.4 Limitations

The evaluation results have shown that the initial hypothesis was confirmed: the metaphor of projecting event logs on maps was sufficiently intuitive and allowed the participants to draw insightful conclusions. The concept and the design of the tool enable users to obtain useful process-related insights. The metaphor of maps and movies is clearly understood and allows stakeholders to gain insights about their process trends and performance, thus guiding the direction for further process mining analysis.

We acknowledge that the scope of the evaluation can definitely be broadened. In particular, it would be worthwhile conducting a thorough study on the effectiveness and efficiency of our tool in comparison with other tools and approaches. Nonetheless, measuring the effectiveness and efficiency of a tool is far from being trivial. The insights that can be gained are subjective by nature. There is no definite answer as to whether the insights obtained through the use of the tool are complete and/or non-trivial. This is also due to the fact in our experiment, none of the subjects involved had ever been involved in a process analysis of this type. Future work includes a comparison of our framework with alternative approaches to verify quantitatively and qualitatively whether our framework is more effective.

We also acknowledge that the number of participants is rather low. As mentioned earlier, given intensive nature of the evaluation method adopted, it is difficult to run this type of evaluations with many users. In our case, each additional subject would require 30–35 min more, from the moment in which the person is introduced to the tool until he/she leaves and is replaced by the subsequent subject. As documented in [8], past applications of methodologies used during our evaluation have shown that a careful choice of experiment subjects, even if relatively small number, can reduce the problem of obtaining subjective results.

Last but not least, it is also worthwhile conducting an evaluation to assess the complexity of manually creating a new map, i.e. without using the *Automatic Map Generator* component.

Despite the issues listed above, we believe that the evaluation has clearly assessed the validity of the framework and the implemented tool. Overall, subjects were enthusiastic about the ability to generate movies based on the event data. Several participants expressed a desire to start using the plug-in in their own analyses. In fact, the tool has helped some end-users find interesting patterns in various situations, some of which were quite surprising to them. Therefore, we conclude that the concept of our visualisation framework is promising and that the current tool already illustrates its potential.

6 Related work

The approach presented in this paper connects two disciplines. On the one hand, we build on techniques for modelling and analysing business processes, such as *business process management* and *process mining* [32]. On the other hand, we build on insights from *visual analytics* [16,31] which are typically not process-aware.

Section 6.1 provides an overview of visualisation-related research in the fields of business process management and process mining. The common denominator of all research work is that they provide static representations (i.e. "photographs" rather than "movies"). When larger data sets come into play, static representations may become problematic as they require large screens to represent the time axis. Conversely, dynamic representations (which exploit physical time) show their power when the focus is on analysing data over time. In general, there is no significant research work that uses dynamic representations, as also stated in [14]: *A frequent goal is to integrate data from multiple time stamps in a single image*. Often, time abstraction is used for this purpose. Unfortunately, to visualise process-related data, temporal aspects are of crucial importance as they are related to concurrency and causality of the activities being performed. Therefore, abstracting time would relegate the time dimension to second-class status. Moreover, the amount of data to visualise would hamper the readability of the diagram, causing it to become useless.

Section 6.2 provides a review of the most significant work in the field of visual analytics. Most of the work in the visual analytics area is tailored to a particular application or to a particular visualisation. In particular, we note that visual analytics has not focused on providing support for the purposes of process analysis. For example, in [1], it is stated that "*To our knowledge, there exists no visualisation framework that […] provides a broader selection of possible representations. We think that an open framework fed with pluggable visual and analytical components for analysing time-oriented data is useful. Such a framework will be able to support multiple analysis tasks and data characteristics, which is a goal of visual analytics*." This is also confirmed by a very detailed up-to-date survey [14].

Note that our framework, which supports the configuration of different maps, allows one to plug-in different representations, thus supporting multiple views.

6.1 Visualisation in business process management and mining

There have been many visualisation techniques proposed in the field of business process management in general, and process mining in particular. However, they are mostly concerned with presenting *static* visualisation of processes (such as process models or time series graphs). Techniques that allow the dynamic representation (e.g. via animation) of processes to highlight how "things" change over time have been sparse. Conversely, one can observe that the research community in the area of information visualisation has not focussed on process-related aspects (examples of such visualisations include [5,14,29]).

We acknowledge that visualisation is quite integral to the field of process mining. Many case studies in the application of process mining techniques in various domains have highlighted the usefulness of visualisation for extracting useful insights from the processes in question (e.g. [9,21,25,30]). Typical visualisations used in the domain of process mining are process models and time series graphs.

Process models are used to visualise patterns of events and their temporal dependencies as learned from event logs. Many approaches to representing historical data as process models have been proposed in this area, such as [4,33,34,36]. In our visualisation framework, instead of resorting to the use of static models to capture generalised temporal patterns of events, we visualise these patterns with a dynamic visualisation where the temporal aspect is a first-class citizen.

Time series graph is another visualisation technique that has been used in the domain of process mining. An example of such a technique is provided by the *basic performance*

analysis ProM plug-in.[14] This plug-in aims to show the performance of processes through a time series view: the x-axis represents a timeline, while the y-axis represents an aggregated amount of working or waiting time for each of the corresponding periods of time. Similarly, the *dotted chart analysis* visualisation technique [28] allows one to visualise events as dots that are spread over time on the x-axis. The y-axis determines how events are grouped within each line, such as per case ID, per activity name, and so forth. Interesting insights can be obtained through dotted chart analysis, such as the time between events, the total case duration as well as the degree of structuredness of the underlying process.[15]

Nevertheless, unlike the work proposed in this article, the visualisation techniques described above are static. Static time series visualisation suffers from some drawbacks when dealing with large amounts of events over a long period of time, namely the impossibility of visualising the time series on a screen in case of huge data sets, even using extremely large screen.

The use of movies to extract knowledge a posteriori from event logs was first explored using the so-called *Fuzzy Animations* [32]. These focus on providing a graphical user interface where the past states extracted from the event log are projected onto a process model which is automatically derived from the events in the log. This feature is also available in a number of commercial business intelligence, such as Disco,[16] Perceptive Process[17] and Celonis.[18] The approach is demonstrated to be effective and similar to our approach. However, it shares the same drawback as many visual analytics approaches: seen from our framework viewpoint, they only support one type of map, the process model.

Other visualisation techniques to support Business Process Management are also proposed. For instance, works [2, 13,26] report on different metaphors and also mash-up approaches to represent the state of a process instance while being executed. Nonetheless, their scopes (and also challenges) are different since they aim at providing run-time decision support on the next activity to work. As a matter of fact, we also proposed an approach for providing run-time decision support to project the active activity instances at run-time over process maps (cf. [7]). The clear difference is that always a kind of photograph referring to the current state is visualised, instead of obtaining a movie.

[14] http://www.processmining.org/online/basicperformanceanalysis.

[15] This can be observed by the presence (or absence) of patterns in the placement of dots.

[16] http://www.fluxicon.com/disco.

[17] http://www.perceptivesoftware.com/products/perceptive-process.

[18] http://www.celonis.de/en/.

The Oracle Business Intelligence (BI) tool[19] supports geospatial visualisation. Again, the main drawback is that, using our terminology, only geographical maps are supported, similarly to the Australian Map in Sect. 4.1.1.

6.2 Visual analytics

Outside the domain of process mining, the use of visual analytics is of course not new. The term "visual analytics" was coined in [31]. This reference reviews the early work in this field. A comprehensive more recent reference is provided in [16]. Examples of recent significant research in the area of visual analytics can be found in document analysis [24], financial analysis [17,37], and geospatial object analysis [3]. In [15], pioneering work is reported where visual analytics is applied to the field of data mining.

Many authors proposed work on the visualisation and analysis of time-oriented data, such as Gantt charts [10]. For time-oriented data, the temporal aspects are considered very important and, hence, should have a strong emphasis when that data are visualised. Event data are time-oriented as each event in the log is associated with a timestamp and it is crucial that this association is clearly visualised. In [18], a research work is reported on where multiple time series are displayed without collapsing (or aggregating) the time aspect into a single frame. Other work, such as Lifelines2 [35] and EventFlow [23], attempts to provide a visualisation framework to gain insights into temporal relationships between events and to answer time-dependent queries of events (for example, whether an event "X" occurred *while* event "Y" is in progress). Nevertheless, the above body of work typically uses static representations for capturing time dependencies, i.e. images that summarise the analysis of time-oriented data.

7 Conclusion

Process mining is a novel discipline that started more than a decade and a half ago. It is concerned with extracting knowledge from event log data to better understand the processes that have been executed within institutions and, subsequently, to improve them. Process mining allows one to discover the actual processes executed within organisations, to pinpoint the deviations with respect to expected or prescribed behaviour and to discover bottlenecks and other performance-related information.

Nowadays, the volume of data that systems produce is growing exponentially. It has been estimated that, today, in 10 mins, the same amount of data is produced as from the stone age till 2013.

[19] http://www.oracle.com/us/bi-enterprise-edition-plus-ds-078848.pdf.

Analysing these data provides new opportunities to understand and improve processes. Dozens of process mining techniques exist depending on the aspects that one wants to look at. Nonetheless, in this "ocean of data", process analysts cannot easily understand where to focus their attention.

This article combines process mining with visual analytics techniques for the purpose of guiding analysts to choose and configure the algorithms that may be effective in their analysis exercises. Existing approaches are unable to represent dynamically what have happened with executed processes. This paper proposes a technique to provide process analysts with a tool to replay the behaviour of executed process instances as recorded in the event log visually. By replaying the event logs, process analysts can gain initial insights into potential problems, which later would need to be confirmed or confuted by applying other process mining techniques.

The basic idea behind our visualisation framework is that process models can be viewed as maps. However, unlike real maps, the quality of process models often leaves much to be desired. Man-made process models tend to be subjective and disconnected from real process executions. Process mining techniques can be used to improve the quality of process maps, e.g. techniques for process discovery can be used to automatically derive process models from event data and conformance checking techniques can be used to pinpoint and quantify deviations between model and reality. However, this is not sufficient as process maps are static and do not show the flow of work. Therefore, we developed an approach to visualise process histories in a generic manner. Any "maps" can be used as long as activity instances can be given coordinates on the maps. For example, an activity instance may be mapped onto a Gantt chart, an organisation chart, and a process model. By showing a sequence of "photographs" of the process in the form of a "movie", one can see concept drift, compliance problems, bottlenecks, etc.

This paper describes an implementation of these ideas in the ProM framework. The ProM implementation is freely available and can be used in conjunction with any event log. To try it out, we made available a sample event log, the corresponding maps, the configuration files, and the related instructions on our website.[20] An initial evaluation of our framework and implementation was conducted using a case study involving a Dutch municipality. The results motivated us to make changes to the framework and the reference implementation. After the adjustments, we performed a second, more extensive evaluation, where Suncorp employees used the tool to analyse event logs. To minimise the risk of obtaining subjective results, we used standard techniques in human–computer interaction to assess the level of usability of the framework's implementation.

[20] http://www.processmining.org/online/logonmaps.

The results in this paper show the value of combining ideas from the fields of process mining and visual analytics. Process mining results are often perceived to be rather abstract and static. Visual analytics approaches tend to be data-centric rather than process-centric. The combination of both fields may yield innovative process-centric visualisations such as the "process movies" proposed in this paper. Note that the maps shown in this paper only serve as examples. One can design maps tailored towards a particular application domain, process, or organisation, thus providing original interactive process visualisations.

Acknowledgments We express our gratitude to Joos Buijs, affiliated with the AIS group of Eindhoven University of Technology, for having performed the initial validation with employees of the Dutch Municipality. This work is partially supported by ARC Discovery Grant DP110100091 "Risk-Aware Business Process Management". Part of this work was conducted when Dr. de Leoni was also affiliated with University of Padua and supported by the Eurostars-Eureka project PROMPT (E!6696).

8 Appendix: Handling loops

In an event log, a loop manifests when two instances of the same activity type at within the same case c are executed sequentially (possibly intertwined with other instances of other activity types within the same case c). Let us call them $ai^1 = (at, c)$ and $ai^2 = (at, c)$. Let us also assume without losing generality that these activity instances enter the *Executing* and *Complete* states, only. Let e_s^1 and e_s^2 be the events when ai^1 and ai^2 starts, respectively. Let e_c^1 and e_c^2 be the events when ai^1 and ai^2 completes, respectively. They are executed sequentially and hence, $timestamp(e_s^1) < timestamp(e_c^1) < timestamp(e_s^2) < timestamp(e_c^2)$. It results in an execution history H as follows:

$$H = \langle \ldots, (S_s^1, timestamp(e_s^1)), \ldots, (S_c^1, timestamp(e_c^1)), \ldots, (S_s^2, timestamp(e_s^2)), \ldots, (S_c^2, timestamp(e_c^2)), \ldots \rangle$$

Let us pick an arbitrary (S, t) in H. We show that the projection of $S = (\alpha, \nu, \tau^s)$ on any map is not influenced by the fact that we do not differentiate ai^1 and ai^2.

With $timestamp(e_s^1) > t$. This case is straightforward: without distinguishing, instance (at, c) is not active yet, thus not visualised; with distinguishing, instances ai^1 and ai^2 are not active, thus not visualised.

With $timestamp(e_s^1) \leq t < timestamp(e_c^1)$. It follows that $\alpha(at, c) = Executing$ and $\tau^s(at, c) = timestamp(e_s^1)$. If it were possible to distinguish ai^1 and ai^2, it would be $\alpha(ai^1) = Executing$ and $\tau^s(ai^1) = timestamp(e_s^1)$ as well as both $\alpha(ai^2)$ and $\tau^s(ai^2)$ would be undefined. Without distinguishing, activity instance (at, c) is visualised; with distinguishing, ai^1, i.e. (at, c), is visualised

with state *Executing*. The definition of the ν function is the same in both situations, as well as $\tau^s(a, c) = \tau^s(ai^1)$. Therefore, the position is the same at which to project ai^1 or (at, c), for the distinguishing and not-distinguishing situation, respectively.

With $timestamp(e_s^1) \leq t < timestamp(e_s^2)$. It follows that $\alpha(at, c) = Completed$ and $\tau^s(at, c)$ is undefined. If it were possible to distinguish ai^1 and ai^2, it would be $\alpha(ai^1) = Completed$ with $\tau^s(ai^1)$ undefined as well as both $\alpha(ai^2)$ and $\tau^s(ai^2)$ would be undefined. Thus, neither activity instance ai^1, if distinguishing, nor (at, c), if not distinguishing, is visualised.

With $timestamp(e_c^2) \leq t < timestamp(e_c^2)$. It follows that $\alpha(at, c) = Executing$ and $\tau^s(at, c) = timestamp(e_s^1)$. If it were possible to distinguish ai^1 and ai^2, it would be $\alpha(ai^1) = Completed$ and with $\tau^s(ai^1)$ undefined as well as both $\alpha(ai^2) = Executing$ and $\tau^s(ai^2)$ would be undefined. Without distinguishing, activity instance (at, c) is visualised; with distinguishing, ai^2, i.e. (at, c), is visualised with state *Executing*. Activity instance ai^1 is not visualised since its state is completed. The definition of the ν function is the same in both situations, as well as $\tau^s(a, c) = \tau^s(ai^1)$. Therefore, the position is the same at which to project ai^2 or (at, c), for the distinguishing and not-distinguishing situation, respectively.

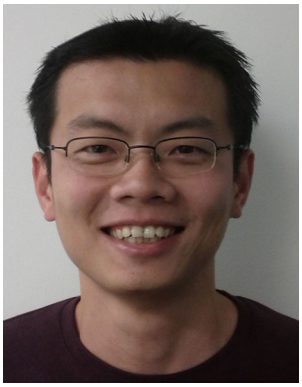
With $t > timestamp(e_c^1)$. It follows that $\alpha(at, c) = Completed$ and $\tau^s(at, c)$ is undefined. If it were possible to distinguish ai^1 and ai^2, it would be $\alpha(ai^1) = \alpha(ai^2) = Completed$ with $\tau^s(ai^1)$ and $\tau^s(ai^2)$ undefined. Thus, neither activity instance ai^1, if distinguishing, nor (at, c), if not distinguishing, is visualised.

References

1. Aigner, W., Miksch, S., Müller, W., Schumann, H., Tominski, C.: Visualizing time-oriented data-a systematic view. J. Comput. Graph. **31**(3), 401–409 (2007)
2. 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, Lecture Notes in Computer Science, vol. 1262, pp. 238–258. Springer, Berlin (1997)
3. Bak, P., Mansmann, F., Janetzko, H., Keim, D.: Spatio-temporal analysis of sensor logs using growth ring maps. IEEE Trans. Vis. Comput. Graph. **15**, 913–920 (2009)
4. Blum, T., Padoy, N., Feussner, H., Navab, N.: Workflow mining for visualization and analysis of surgeries. CARS J. **3**, 379–386 (2008)
5. Chen, C.: Information Visualization: Beyond the Horizon. Springer, New York, Inc (2006)
6. de Leoni, M., Buijs, J.C.A.M, van der Aalst, W.M.P., ter Hofstede, A.H.M.: Facilitating process analysis through visualising process history: experiences with a Dutch municipality. Technical report, BPM Center Report BPM-12-24, BPMcenter.org (2014)
7. de Leoni, M., Adams, M., van der Aalst, W.M.P., ter Hofstede, A.H.M.: Visual support for work assignment in process-aware information systems: framework formalisation and implementation. Decis. Support Syst. **54**(1), 345–361 (2012)
8. Dix, A., Finlay, J.E., Abowd, G.D., Beale, R.: Human–Computer Interaction, 3rd edn. Prentice Hall, Englewood Cliffs NJ (2003)
9. Fitzgerald, J.A., Dadich, A.: Using visual analytics to improve hospital scheduling and patient flow. J. Theor. Appl. Electron. Commer. Res. **4**(2), 20–30 (2009)
10. Gantt, H.L.: A graphical daily balance in manufacture. ASME Trans. **24**, 1322–1336 (1903)
11. Günther, C.W., van der Aalst, W.M.P.: Fuzzy mining: adaptive process simplification based on multi-perspective metrics. In: BPM 2007, LNCS, vol. 4714, pp. 328–343. Springer, Berlin (2007)
12. Heer, J., Robertson, G.: Animated transitions in statistical data graphics. IEEE Trans. Vis. Comput. Graph. **13**(6), 1240–1247 (2007)
13. 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. Environ. Model. Softw. **20**(6), 689–704 (2005)
14. Kehrer, J., Hauser, H.: Visualization and visual analysis of multifaceted scientific data: a survey. IEEE Trans. Vis. Comput. Graph. **19**(3), 495–513 (2013)
15. Keim, D.: Visual exploration of large data sets. Commun. ACM **44**, 38–44 (2001)
16. 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)
17. 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), pp. 195–200. Eurographics Association (2006)
18. Krstajic, M., Bertini, E., Keim, D.A.: Cloudlines: Compact display of event episodes in multiple time-series. IEEE Trans. Vis. Comput. Graph. **17**(12), 2432–2439 (2011)
19. Laney, D., Bitterer, A., Sallam, R., Kart, L.: Predicts 2013: Information innovation. Gartner (G00246040) (2012)
20. Li, J., Martens, J.B., van Wijk, J.J.: A model of symbol size discrimination in scatterplots. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10), pp. 2553–2562. ACM (2010)
21. Mans, R.S., Schonenberg, M.H., Song, M., van der Aalst, W.M.P., Bakker, P.J.M.: Application of Process Mining in Healthcare - a case study in a Dutch Hospital. In: Biomedical Engineering Systems and Technologies, Communications in Computer and Information Science, vol. 25, pp. 425–438. Springer, Berlin, Heidelberg (2009)
22. Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., Byers, A.H.: Big Data: The Next Frontier for Innovation, Competition, and Productivity. McKinsey Global Institute (MGI) (2011)
23. Monroe, M., Lan, R., Lee, H., Plaisant, C., Shneiderman, B.: Temporal event sequence simplification. IEEE Trans. Vis. Comput. Graph. **19**(12), 2227–2236 (2013)
24. 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), pp. 187–194. IEEE (2009)
25. Pechenizkiy, M., Trčka, N., Vasilyeva, E., van der Aalst, W.M.P., Bra, P.D.: Process Mining Online Assessment Data. In: Romero, C., Ventura, S., Pechenizkiy, M., Baker, R. (eds.) Educational Data Mining (EDM 2009), pp. 279–288. www.educationaldatamining.org (2009)
26. Schönhage, B., Eliëns, A.: Management through vision: a case study towards requirements of BizViz. In: AVI 2000: International Conference of Information Visualisation, pp. 387–392. IEEE Computer Society (2000)

27. Simkin, D., Hastie, R.: An information-processing analysis of graph perception. J. Am. Stat. Assoc. **82**(398), 454–465 (1987)
28. Song, M., van der Aalst, W.M.P.: Supporting process mining by showing events at a glance. In: Chari, K., Kumar, A. (eds.) Proceedings of the 17th Annual Workshop on Information Technologies and Systems (WITS 2007), pp. 139–145. Montreal, Canada (2007)
29. Spence, R.: Information Visualization: Design for Interaction, 2nd edn. Pearson Education Limited, Harlow, England (2006)
30. Suriadi, S., Wynn, M., Ouyang, C., ter Hofstede, A.H.M., van Dijk, N.: Understanding process behaviours in a large insurance company in Australia : a case study. In: Salinesi, C., Norrie, M., Pastor, O. (eds.) International Conference on Advanced Information Systems Engineering, LNCS, vol. 7908, pp. 449–464. Springer, Heidelberg (2013)
31. Thomas, J.J., Cook, K.A. (eds.): Illuminating the Path: The Research and Development Agenda for Visual Analytics. IEEE CS Press, Silver Spring, MD (2005)
32. van der Aalst, W.M.P.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer, Berlin Heidelberg (2011)
33. van Dongen, B.F., Adriansyah, A.: Process mining: fuzzy clustering and performance visualization. In: Business Process Management Workshops. LNBIP, vol. 43, pp. 158–169 Springer, Berlin, Heidelberg (2009)
34. Verbeek, H.M.W., Pretorius, A., van der Aalst, W.M.P., van Wijk, J.J.: On petri-net synthesis and attribute-based visualization. In: Moldt, D., Kordon, F., Hee, K., Colom, J., Bastide, R. (eds.) Proceedings of the Workshop on Petri Nets and Software Engineering (PNSE'07), pp. 127–142. Publishing House of University of Podlasie, Siedlce, Poland (2007)
35. Wang, T., Plaisant, C., Shneiderman, B., Spring, N., Roseman, D., Marchand, G., Mukherjee, V., Smith, M.: Temporal summaries: supporting temporal categorical searching, aggregation and comparison. IEEE Trans. Vis. Comput. Graph. **15**(6), 1049–1056 (2009)
36. Weijters, A.J.M.M., van der Aalst, W.M.P., de Medeiros, A.A.: Process Mining with the Heuristics Miner-algorithm. BETA Working Paper Series, WP 166, Eindhoven University of Technology, Eindhoven, The Netherlands (2006)
37. 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), pp. 19–26. Eurographics Association (2007)

Massimiliano de Leoni is an Assistant Professor of Information Systems at the Technische Universiteit Eindhoven (TU/e), the Netherlands. In 2009, he earned a PhD in Computer Engineering at SAPIENZA—University of Rome, Italy, on "Adaptive Process Management in Highly Dynamic and Pervasive Scenarios". He was a guest research fellow at Queensland University of Technology, Vienna University of Economics, and Business and University of Naples. His research interests are in the areas of Process-aware Information Systems and Business Process Management, predominantly on multi-perspective process mining, process-aware decision support systems as well as on visualization techniques for business process management and analysis.



Suriadi Suriadi is currently a Research Fellow within the Business Process Management Discipline of Queensland University of Technology, Brisbane, Australia. He obtained his PhD degree in the discipline of Information Security in late 2010 from the Queensland University of Technology (QUT). Since 2007, he has been involved in a number of research projects both in the area of information security and business process management. He enjoys working in collaborative, cross-domain research projects that allow the application of research outcomes to address real-world problems. His main research interests are in the area of process mining and information security.

Arthur H. M. ter Hofstede is a Professor in the Information Systems School in the Science and Engineering Faculty, Queensland University of Technology, Brisbane, Australia, where he is Head of the Business Process Management Discipline. He is also a Professor in the Information Systems Group of the Department of Industrial Engineering of the Technische Universiteit Eindhoven (TU/e). His research interests are in the areas of business process automation and process mining.

Wil M. P. van der Aalst is a Full Professor of Information Systems at the Technische Universiteit Eindhoven (TU/e), the Netherlands. He is also the Academic Supervisor of the International Laboratory of Process-Aware Information Systems of the National Research University, Higher School of Economics in Moscow. Moreover, since 2003, he has a part-time appointment at Queensland University of Technology (QUT). At TU/e he is the Scientific Director of the Data Science Centre Eindhoven (DSC/e). Wil van der Aalst has published more than 165 journal papers, 17 books (as author or editor), 350 refereed conference/workshop publications, and 60 book chapters. Many of his papers are highly cited (he has an H-index of more than 110 according to Google Scholar) and his ideas have influenced researchers, software developers, and standardization committees working on process support. In 2012, he received the degree of doctor honoris causa from Hasselt University. In 2013, he was appointed as Distinguished University Professor of TU/e and was awarded an honorary guest professorship at Tsinghua University. He is also a member of the Royal Holland Society of Sciences and Humanities (Koninklijke Hollandsche Maatschappij der Wetenschappen) and the Academy of Europe (Academia Europaea).