Discovering High-Level Performance Models for Ticket Resolution Processes
(Short Paper)

Francesco Folino, Massimo Guarascio, and Luigi Pontieri

Institute ICAR, National Research Council of Italy (CNR)
Via Pietro Bucci 41C, I87036 Rende (CS), Italy
{ffolino,guarascio,pontieri}@icar.cnr.it

Abstract. Predicting run-time performances is a hot issue in ticket resolution processes. Recent efforts to take account for the sequence of resolution steps, suggest that predictive Process Mining (PM) techniques could be applied in this field, if suitably adapted to the peculiarities of ticket systems. In particular, the performances of a ticket instance usually depend on which kinds of experts worked on it (more than on the mere sequence of resolution tasks), while relevant information about ticket cases is stored in the form of text fields, which are usually disregarded by PM approaches. Instead of relying on a-priori experts groups, we devise an ad-hoc method for clustering experts according to their real working patterns, based on log data. Regarding the discovered groups as abstractions for log events, we also perform a predictive clustering of ticket cases, while using context data as input attributes for splitting the tickets. In this way, different (context-dependent) execution scenarios are recognized for the process, and equipped with more accurate performance predictors. The approach was validated on a real application scenario, where it showed better results than state-of-the-art solutions.

1 Introduction

Ticket management systems (devoted to manage, maintain, and help resolve issues in an organization) are very popular in various collaboration environments (such as, e.g., CSCW, Social Computing, Bug Tracking systems), and they have been given increasing attention recently, especially as concerns the opportunity to exploit their log data to improve the handling of tickets. So far, most efforts have been spent to extract some kind of recommendation model for improving the allocation of (human) resources. To this end, some approaches just look at the text information (e.g., keywords, notes) stored for each ticket [3], while others also look at the sequence of worker groups each ticket has been progressively assigned to [7,10]. In general, an implicit goal of all the approaches above is to keep as short as possible the length and/or the duration of ticket resolution sequences, while possibly optimizing the usage of resources (i.e., experts or "workers"). Predicting process instances' performances is right the scope of an emerging research stream in the field of Process Mining [1]. The basic idea is to exploit historical

R. Meersman et al. (Eds.): OTM 2013, LNCS 8185, pp. 275–282, 2013.

log data, in order to induce a state-aware performance model for the analyzed process, based on some log abstraction function (see, e.g., [2, 5]). A key point in the induction of predictive performance models (particularly when working with a ticket management process) is right the definition of a suitable log abstraction function. To this regard, most Process Mining approaches tend to rely on activity labels, assuming that they capture well the semantics of each processing step. However, most ticket systems just support generic activities, such as, e.g., open/close/lock/unlock a ticket, bounce/forward it, send a message, add comment, update a ticket's descriptor. Considering the intervention of workers can help gain insight into the resolution stage of a ticket, but looking at the very individuals involved is likely to yield unreadable and imprecise (overfitting) models. On the other hand, the idea of relying on pre-defined worker groups (as in [7, 9, 10]) may well be unsuitable for many real application contexts, where such groups do not exist or do not reflect the behavior of staff people.

In this work, we propose an approach to the prediction of performances for ticket resolution instances, which overcomes the limitations discussed above, while taking account for both context data and resolution steps. Specifically, in order to get a suitable abstraction level over the performed steps, we only focus on the sequence of workers involved in a ticket, abstracted into high-level groups. An ad-hoc clustering method is devised to discover such groups out of log data (while possibly reusing a-priori knowledge), so that they reflect real different resolution skills and performance profiles. These data-driven groups are the basis for regarding the sequence of events recorded in each ticket-resolution trace at a higher level of abstraction (i.e., as a sequence of worker groups), before inducing a predictive model for the process. Technically, this latter task can be accomplished by integrating a classic state-aware performance prediction method [2] into a predictive clustering scheme [4], as proposed in [5]. Such a clustering of ticket traces can help detect different (context-dependent) ticket handling cases, and equip them all with separate and precise performance predictor. Preliminary results on real-life data show that enhancing this latter approach with our clustering-based event abstraction method allows to build more effective models.

2 Preliminaries

Generally, since each ticket gives rise to a series of activities, it can be viewed as an instance of some (ticket handling) process. We then assume that a record (named *trace*) is kept for each ticket, which stores the sequence of *events* occurred during the corresponding process instance – where each event regards the execution of some activity. In the rest of the paper, we will only focus on manual activities performed by some *worker* (i.e., a member of the staff), including those (usually named "experts") who are in charge of finding a solution to the signaled issue. With little language abuse, we will use "trace", and "ticket" interchangeably, as the former is a direct consequence of the creation of a ticket.

Let $A = \{a_1, \ldots, a_{\#A}\}$ be the tasks composing the (ticket handling) process, $W = \{w_1, \ldots, w_{\#W}\}$ be the workers who can perform them, E be the universe of

all events that can happen during the handling of tickets, and $\mathcal{T}$ be the universe of all possible ticket traces. Moreover, for any event $e \in E$, let $task(e)$, $executor(e)$, and $time(e)$ be the activity, executor and timestamp, resp., associated with e.

For each trace $\tau \in \mathcal{T}$, let $len(\tau)$ be the number of events stored in τ, let $\tau[i]$ be the i-th event of τ, for $i = 1 \,..\, len(\tau)$, and $\tau(i] \in \mathcal{T}$ be the *prefix* trace consisting only of the first i events in τ, for any $i = 0, \ldots, len(\tau)$. Finally, a *log* L (over $\mathcal{T}$) is a finite subset of $\mathcal{T}$, while the *prefix set* of L, denoted by $\mathcal{P}(L)$, is the set of all prefix traces that can be extracted from L.

In addition to resolution steps, ticket management systems usually keep several descriptive fields for each ticket instance. Most systems store, for each ticket, several kinds of free-text contents, ranging from notes to messages. Let us assume that the textual contents of any ticket τ is conveyed by a vector $text(\tau)$, encoding a bag-of-word representation of the concatenation of all τ's free-text fields – to this aim, classic vector-space model and TF-IDF weighting are used. Moreover, like in [5], we assume that a vector $context(\tau)$ is associated with any ticket trace τ, encoding all context data available for τ. Note that, besides "intrinsic" properties (explicitly stored for each ticket), a series of "extrinsic" context features (such as, e.g., workload indicators, temporal dimensions), can be derived for each trace, capturing the state of the ticket system when it began.

Performances: Measures and Models. Let us assume that $\hat{\mu} : \mathcal{T} \to \mathbb{R}$ is an (unknown) function assigning a performance value to any (possibly partial) process instance. Then, a (predictive) *Process Performance Model* (*PPM*) is a model that can forecast the performance value of any process instance, based on the events happened during its enactment. Such a model can be represented as a function $\mu : \mathcal{T} \to \mathbb{R}$, estimating $\hat{\mu}$ all over the trace universe. Learning a PPM is a special induction problem, where the training set is a log L, and $\hat{\mu}(\tau)$ is known for each (sub-)trace $\tau \in \mathcal{P}(L)$. Recent approaches to the induction of a PPM (e.g., [2, 5]) share the idea of regarding process logs at a suitable level of abstraction, with the help of functions like those described next.

An *event abstraction function* $\mathcal{E} : E \to \mathcal{A}^E$ is a function mapping each event $e \in E$ to an abstract representation $\mathcal{E}(e)$ in some proper space $\mathcal{A}^E$. Moreover, a *trace abstraction function* $abs_{\mathcal{E}} : \mathcal{T} \to \mathcal{A}^{\mathcal{T}}$, w.r.t. an event abstraction function $\mathcal{E}$, is a function mapping each trace $\tau \in \mathcal{T}$ to an abstract representation $abs_{\mathcal{E}}(\tau)$ in some space $\mathcal{A}^{\mathcal{T}}$. Let us consider the three trace abstraction functions (practically equivalent to those introduced in [2]) $list^h_{\mathcal{E}}(\tau)$, $bag^h_{\mathcal{E}}(\tau)$, and $set^h_{\mathcal{E}}(\tau)$, which map any trace τ to the *list*, *multiset*, or *set*, respectively, of the h latest events occurring in τ, while abstracting each of these events via function $\mathcal{E}$. Based on such trace abstractions, a PPM is derived in [2] in terms of an annotated finite state machine (named "A-FSM"), where: *(i)* each node corresponds to one abstract trace representation α (produced by $abs_{\mathcal{E}}$), and stores an estimate for the target measure (usually computed as the average over all the trace prefixes matching α), while *(ii)* each transition is labelled with an event abstraction (produced by $\mathcal{E}$). This learning approach was hybridized in [5] with a predictive clustering scheme [4], where context information are used as descriptive features while the targets are derived from performance measurements. As discussed before, the

event abstraction functions used in [2, 5] (and in current literature generally) are pretty simple, for they just select some properties of an event e – typically, $executor(e)$, $task(e)$ or both – and risk being ineffective in the case of tickets.

Example 1. Let us introduce a toy example, inspired to the real-life logs considered in Section 5. Let τ and τ' be two traces storing the event sequences $\langle e_1, e_2, e_3, e_4, e_5\rangle$ and $\langle e'_1, e'_2, e'_3, e'_4, e'_5\rangle$, respectively, with: $executor(e_1)$=$system$, $executor(e_2)$=$executor(e_3)$=$john$, $executor(e_4)$=$mark$, $executor(e_5)$=$system$; $executor(e'_1)$=$system$, $executor(e'_2)$=$mark$, $executor(e'_3)$=$executor(e'_4)$=$john$, $executor(e'_5)$=$system$. Let also $\mu(\tau(1]) = 6$, $\mu(\tau(2]) = 4$, $\mu(\tau(3]) = 2$, $\mu(\tau(4]) = 1$, $\mu(\tau(5]) = 0$; $\mu(\tau'(1]) = 4$, $\mu(\tau'(2]) = 3$, $\mu(\tau'(3]) = 2$, $\mu(\tau'(4]) = 1$, $\mu(\tau'(5]) = 0$. Consider, as a straightforward event abstraction criterion, a function γ mapping each event e to $executor(e)$. It is easy to see that for the prefix $\tau(3]$ it is: $list^{\infty}_{\gamma}(\tau(2]) = \langle system, john, john\rangle$, $bag^{\infty}_{\gamma}(\tau(2]) = [system, john^2]$, $set^{\infty}_{\gamma}(\tau(3]) = \{system, john\}$. Conversely, when $h = 1$, all abstractions $list^1_{\gamma}(\tau(3])$, $bag^1_{\gamma}(\tau(3])$ and $set^1_{\gamma}(\tau(3])$ will just consist of the sole element $john$, which is indeed the last executor appearing in $\tau(3]$. The A-FSM model derived from the log $\{\tau, \tau'\}$ with $list^2_{\gamma}$ will contain the states $\langle system\rangle$, $\langle system, john\rangle$, $\langle john, john\rangle$, $\langle john, mark\rangle$, $\langle mark, system\rangle$, $\langle system, mark\rangle$, $\langle mark, john\rangle$, and $\langle john, system\rangle$, annotated with the performance estimations 5, 4, 3, 1, 0, 3, 2, and 0, respectively.

3 A Clustering Framework for the PPM Discovery

We next define the specific kind of PPM model we want to learn, extending the notion of "Context-Aware PPM" of [5], with the capability of gaining an effective abstraction level over ticket traces.

A *context-aware process performance model* (`HO-PPM`) for a given log L is a triple of the form $M = \langle abs, part, \langle \mu_1, \ldots, \mu_k\rangle\rangle$, such that: *(i)* abs is a trace abstraction function, *(ii)* $part$ is a trace partitioning function allowing to split log traces into a number, say k, of disjoint clusters, based on their context data, and *(iii)* μ_i is an A-FSM, for $i = 1, \ldots, k$, where the states are labelled with the abstract representations produced by abs. This model encodes a predictive clustering function estimating the unknown performance function $\hat{\mu}$ as follows: $\mu^M(\tau)$=$\mu_j(abs(\tau))$ such that j=$part(context(\tau))$. In this way, a forecast for any new process instance τ is made by first assigning τ to a cluster (via function $part$, based on $context(\tau)$), and then providing the predictor of that cluster with the abstract representation of τ.

In order to make this model really effective, the event abstraction function must be defined carefully. To this end, we automatically partition workers into performance-relevant groups, through an ad-hoc clustering procedure, while exploiting log information, as a reflection of real workers' behaviors. These groups will then be regarded as high-level representations of log events, by technically defining an event abstraction function which maps each event e to the cluster of the worker associated with e, i.e., with $executor(e)$.

We next illustrate how a dissimilarity measure for workers can be defined, in order to eventually partition them into performance-relevant groups, with the

help of whatever distance-based clustering method. Essentially, a rough performance indicator is defined, named *performance footprint*, to capture the impact of a worker on the performances of the tickets she participated to. For significance reasons, footprints are defined in an aggregated way, looking at the participation of workers to a given set of ticket classes, denoted by $C^{\mathcal{T}} = \{c_1^{\mathcal{T}}, \ldots, c_{\#C}^{\mathcal{T}}\}$.

For any ticket trace τ and any worker w appearing in it (as an executor), let $perf(w, \tau)$ be the performance value corresponding to the the last τ's event associated with w. Further, for any ticket class $c \in C^{\mathcal{T}}$ and any worker w, let $c(w)$ be the tickets of c including w. Then, the *performance footprint* of w w.r.t. c is defined as follows: $pf(w, c) =$ `NULL` if $|c(w)| <$ `MIN_COUNT`, or $\sum_{\tau \in c(w)} perf(w, \tau)/|c(w)|$, otherwise. Note that in the tests shown later on, `MIN_COUNT` (minimal nr. of samples for computing $pf(w, c)$) was set to 50 – but a wider range of values worked fine as well. As an instance, let c be a class gathering the two example traces τ and τ' of Example 1, while assuming that `MIN_COUNT` $= 0$. It is easily seen that $perf(john, \tau) = \mu(\tau(3]) = 2$, $perf(john, \tau') = \mu(\tau'(4]) = 1$, whereas $perf(mark, \tau) = \mu(\tau(4]) = 1$, $perf(mark, \tau') = \mu(\tau'(2]) = 3$. Then, it is $pf(john, c) = (1 + 2)/2 = 1.5$ and $pf(mark, c) = (1 + 3)/2 = 2$.

Often, background knowledge is available concerning the relationships between workers and organization-oriented concepts (such as roles, teams, capabilities and skills). W.l.o.g., we assume that such information is encoded through a function org_O, mapping each worker w to the set $org_O(w)$ of organizational concepts she is associated with. Hereinafter, we will call $org_O(w)$ the (a-priori) *organizational profile* of worker w, and org_O an *organizational-profiling* function.

We can now define the overall distance measure for partitioning the executors of ticket management activities into disjoint clusters, each corresponding to some specific combination of capabilities and performance profiles.

Definition 1 (Workers' Distance). Let L be a ticket management log, $C^{\mathcal{T}}$ be a partitioning of all the tickets in L. Moreover, let $org_O : W \rightarrow 2^O$ be an organizational-profiling function (w.r.t. a given set O of organizational concepts). Then, the distance $dist^W(w_1, w_2)$ between two any workers $w_1, w_2 \in W$ (w.r.t. org_O, L and $C^{\mathcal{T}}$) is computed as a linear combination of three functions: $dist^W(w_1, w_2) = \beta \times dist_O(w_1, w_2)) + (1-\beta) \times [dist_A(w_1, w_2)) + dist_P(w_1, w_2)]/2$. Function $dist_O(w_1, w_2))$ (resp., $dist_A(w_1, w_2))$) simply corresponds to the Jaccard distance between the sets of organizational concepts associated with (resp., of tasks executed by) w_1 and w_2, while the third (footprint-based) distance is defined as follows: $dist_P(w_1, w_2)) = \sqrt{\sum_{c \in C^{\mathcal{T}}} \delta\left(pf(w_1, c), pf(w_2, c)\right)^2 / |C^{\mathcal{T}}|}$, with $\delta(x, y) = \frac{|x-y|}{\max\{pf(w,c) | w \in W, c \in C^{\mathcal{T}}\}}$ if `NULL` $\notin \{x, y\}$, and $\delta(x, y) = 1$ otherwise; □

4 Computation Approach: Algorithm HOPP

An algorithm, named `HOPP` (i.e., High-Order Performance Prediction), for extracting an `HO-PPM` from a ticket log, is shown in Figure 1. Besides historical log traces (and associated performance measurements), the algorithm takes as input a set of workers, along with their organizational profiles, and an initial partition

Input: A log L over ticket (trace) universe $\mathcal{T}$, a target measure $\hat{\mu}$ known over $\mathcal{P}(L)$, an a-priori partition $C^{\mathcal{T}}$ of tickets in classes, a set O of (organznl.) concepts, a set W of workers, a function $org : W \rightarrow 2^O$, $mode \in \{\texttt{LIST}, \texttt{SET}, \texttt{BAG}\}$, $h \in \mathbb{Z}$, and $\beta \in [0, 1]$.
Output: An HO-PPM model for L (fully encoding $\hat{\mu}$ all over $\mathcal{T}$).
Method: Perform the following steps:
1 Derive $context(\tau)$ for each $\tau \in L$, by computing $text(\tau)$, $data(\tau)$ and $env(\tau)$;
2 Compute $pf(w, c)$ and $tasks_L(w)$ for each $w \in W$ and $c \in C^{\mathcal{T}}$;
3 Compute a clustering C^W of workers, by using the distance in Def. 1;
4 Define an event abstraction function $\gamma : E \rightarrow C^W$ s. t. $\gamma(e)$ is the group including $executor(e)$;
5 Let $abs = list^h_\gamma$ (resp., set^h_γ, bag^h_γ) if $mode = \texttt{LIST}$ (resp., SET, BAG);
6 Let $AS = \{a_1, ..., a_k\}$ be the set of all trace abstractions produced by abs on L – i.e., $AS = \{abs(\tau) \mid \tau \in \mathcal{P}(L)\}$
7 Learn a PCT model T, using $context(\tau)$ (resp., $val(\tau, a_i)$, i=1..k) as descriptive (resp., target) features $\forall \tau \in L$;[a]
8 Let $part_T$ be the partitioning function of T and let $L[1], \ldots, L[q]$ be the clusters obtained by applying $part_T$ to L;
9 **for each** $L[i]$ **do** Induce an A-FSM model μ_i (w.r.t. abs) out of $L[i]$ **end**;
10 **return** $\langle abs, part_T, \langle\ \mu_1, \ldots, \mu_k \rangle\ \rangle$

[a] $val(\tau, \alpha) =$ NULL if $matchs(\tau, \alpha) = \emptyset$, or $val(\tau, \alpha) = \hat{\mu}(\max(matchs(\tau, \alpha)))$, otherwise; where $matchs(\tau, \alpha)$ is the set of τ's prefixes matching a given trace abstraction α.

Fig. 1. Algorithm HOPP

$C^{\mathcal{T}}$ of tickets into classes (possibly capturing different kinds of ticket resolution problems). Parameters *mode* and h let the user choose the trace abstraction for building all A-FSM models, while β controls the relative weight of background knowledge in the clustering of workers.

Notice, in particular, that the (preliminary) partitioning of tickets allows to compute the (aggregated) performance footprints of workers, and to carry out a clustering over the domain of workers. To this end, our current implementation uses algorithm *X-means* [8] with the distance in Def. 1. If no a-priori ticket clustering is given, one can compute it with *X-means* – as done in our tests.

The discovered worker groups are then used to transform the original log, by replacing each log trace with a higher-level version of it (Step 5), where each log event e is mapped to the group of the worker who executed e. This abstracted version of the log is then exploited to extract a HO-PPM, using the context features of tickets as descriptive attribute, and performance values as target features. This task is carried out by following the approach in [5], where the discovery of such a model is accomplished in two phases: first a *Predictive Clustering Tree (PCT)* [4] is induced (Step 7) from a propositional view of the log; then the method in [2] is used to equip each of the discovered clusters with an *A-FSM* model (Step 9) – while using workers groups for event abstraction.

5 Case Study

This section illustrates the results of an experimental activity, performed to asses the validity of the proposed approach. This empirical analysis was carried out on a real-life scenario, concerning the handling of tickets in the IT department of an Italian enterprise. The typical lifecycle of a ticket, in the analyzed ticket

Table 1. Remaining steps' prediction: error reductions (%) of HOPP w.r.t. CA-TP and FSM, when using different trace abstraction functions (while fixing $\beta = 0.5$)

Trace abstraction function		**Δ% w.r.t. CA-TP [5]**			**Δ% w.r.t. FSM [2]**		
mode	*h*	*rmse*	*mae*	*mape*	*rmse*	*mae*	*mape*
LIST	2	-17.8%	-5.4%	-8.4%	-19.1%	-15.3%	-31.0%
	4	-31.2%	-33.4%	-26.4%	-34.1%	-45.8%	-48.4%
	8	-45.7%	-47.2%	-40.4%	-47.7%	-55.5%	-56.4%
	16	-48.2%	-49.5%	-43.5%	-50.0%	-57.2%	-58.9%
	Total	**−35.7%**	**−33.9%**	**−29.6%**	**−37.7%**	**−43.4%**	**−48.7%**
BAG	2	-15.5%	-7.5%	-8.8%	-17.1%	-14.6%	-21.4%
	4	-29.2%	-35.3%	-21.6%	-30.8%	-45.0%	-38.9%
	8	-42.2%	-45.8%	-39.2%	-44.0%	-52.2%	-52.0%
	16	-47.5%	-50.0%	-43.3%	-49.1%	-55.8%	-55.3%
	Total	**−33.6%**	**−34.7%**	**−28.2%**	**−35.2%**	**−41.9%**	**−41.9%**
Grand Total		**−34.7%**	**−34.3%**	**−28.9%**	**−36.5%**	**−42.7%**	**−45.3%**

management system, can be summarized as follows. As soon as a problem is reported, a ticket is created by the help desk with a short description of the problem signaled. The ticket is then sent via email to one expert, based on predefined (and partly implicit) allocation schemes. If the worker solves the problem, she closes the ticket; otherwise, she forwards it to another worker, while possibly making her informed about the handed-over case. Several kinds of data are stored for a ticket, including: the actions performed on it and their executors, a priority and a severity level, and general categories for the reported issue (e.g., logging-in problems, service outage, etc.). Moreover, as to free-text ticket contents, we considered the ticket summary (describing its nature and status) and the subjects and bodies of all messages exchanged between people working on it. These textual contents were summarized into a selection of keywords — precisely, the top 500 stemmed terms w.r.t. IDF scores, occurring in at least 10 tickets. The workers involved in the resolution of tickets were structured in 50 groups. In our tests, we focused on 2286 tickets handled in 2 consecutive months of year 2012. Since no a-priori partitioning $C^{\mathcal{T}}$ of the tickets was given, we computed it by applying *X-means* [8] on ticket data, while using an ad-hoc contents-oriented distance function for comparing them. Specifically, the distance between two tickets was computed by combining the *cosine* distance between the (vector space representations of) their respective text contents, with a mismatch-based distance between their respective data fields. As categorical attributes usually correspond to predefined categories (assigned by help-desk staff), we used the same factor β, as for workers' clustering, to control their influence on distance evaluation — the higher β, the higher the influence.

In order to quantify prediction accuracy, we used the same error metrics as in [2] — namely, *rmse*, *mae*, and *mape* (mean absolute percentage error) — measured via a (10 fold) cross-validation procedure. Table 1 reports the error reduction (in percent, denoted by Δ%) achieved by the algorithm HOPP when it is applied to the dataset described above to predict the resolution steps needed for a ticket w.r.t. two state-of-the-art approaches denoted by CA-TP [5] and FSM [2], respectively. The tests were performed by setting the value of parameter β (i.e., the weight given to background categories in the clustering of workers/tickets)

to 0.5 – although the method seems to be quite robust to value of β taken out of $\{0.25, 0.5, 0.75\}$ – and different kinds of trace abstraction functions, specified through the parameters h and *mode* $\in \{\texttt{LIST}, \texttt{BAG}\}$. Even if the advantage of using our solution are already appreciable with $h = 2$, higher horizon values yield better performances. The effect of the abstraction mode is not very marked, seeing as very similar (good) results are found in both cases. In fact, whatever h, less than 2.5% error reduction is obtained (on all metrics) when moving from bags to lists, for all metrics but the *mape* – which shrinks of nearly 7%.

6 Conclusion

The paper presented an approach to the prediction of performances for ticket resolution instances. The approach features several innovative aspects, including: (i) the exploitation of textual contents (ignored by current process mining methods), and (ii) a data-driven (clustering-based) event abstraction procedure. As to future work, we plan to integrate the approach into a real ticket management system, and to try to make the clusterings of tickets and workers more synergistic – e.g., via some multi-way co-clustering scheme (like that in [6]).

References

1. van der Aalst, W.M.P., van Dongen, B.F., Herbst, J., Maruster, L., Schimm, G., Weijters, A.J.M.M.: Workflow mining: a survey of issues and approaches. Data & Knowledge Engineering 47(2), 237–267 (2003)
2. van der Aalst, W.M.P., Schonenberg, M.H., Song, M.: Time prediction based on process mining. Information Systems 36(2), 450–475 (2011)
3. Anvik, J., Hiew, L., Murphy, G.C.: Who should fix this bug? In: Proc. of 28th Int. Conf. on Software Engineering (ICSE 2006), pp. 361–370 (2006)
4. Blockeel, H., Raedt, L.D.: Top-down induction of first-order logical decision trees. Artificial Intelligence 101(1-2), 285–297 (1998)
5. Folino, F., Guarascio, M., Pontieri, L.: Discovering context-aware models for predicting business process performances. In: Meersman, R., et al. (eds.) OTM 2012, Part I. LNCS, vol. 7565, pp. 287–304. Springer, Heidelberg (2012)
6. Greco, G., Guzzo, A., Pontieri, L.: Co-clustering multiple heterogeneous domains: Linear combinations and agreements. IEEE Transactions on Knowledge and Data Engineering 22(12), 1649–1663 (2010)
7. Miao, G., Moser, L.E., Yan, X., Tao, S., Chen, Y., Anerousis, N.: Generative models for ticket resolution in expert networks. In: Proc. of 16th Int. Conf. on Knowledge Discovery and Data Mining (KDD 2010), pp. 733–742 (2010)
8. Pelleg, D., Moore, A.W.: X-means: Extending k-means with efficient estimation of the number of clusters. In: Proc. of 17th Int. Conf. on Machine Learning (ICML 2000), pp. 727–734 (2000)
9. Shao, Q., Chen, Y., Tao, S., Yan, X., Anerousis, N.: Efficient ticket routing by resolution sequence mining. In: Proc. of 14th Int. Conf. on Knowledge Discovery and Data Mining (KDD 2008), pp. 605–613 (2008)
10. Sun, P., Tao, S., Yan, X., Anerousis, N., Chen, Y.: Content-aware resolution sequence mining for ticket routing. In: Hull, R., Mendling, J., Tai, S. (eds.) BPM 2010. LNCS, vol. 6336, pp. 243–259. Springer, Heidelberg (2010)