MANA: Identifying and Mining Unstructured Business Processes

Pedro M. Esposito[1], Marco A. A. Vaz[1], Sérgio A. Rodrigues[1], Jano M. de Souza[1]

[1] Federal University of Rio de Janeiro, Brazil
{pmesposito, mvaz, sergio, jano}@cos.ufrj.br

Abstract. The process mining field supports the discovery of process models using audit trails logged by information systems. Several mining techniques are able to deal with unstructured processes, mainly through cluster analysis. However, they assume the previous extraction of an event log containing related instances. This task is not trivial when the source system doesn't provide a reliable separation of its processes and allows the input of data through free text fields. The identification of related instances should, in this case, be explorative and integrated into the process mining tool used in later stages of the analyst's workflow. To this goal, the MANA approach was developed, allowing the explorative selection and grouping of instances through a canonical database.

Keywords. Process Mining, Process Discovery, Business Process Management, Unstructured Processes

1 Introduction

Process discovery, one of the main branches of process mining, involves the discovery of process models from event logs extracted from information systems. This approach can avoid the modeling of biased and too broad views of the process [1] and allows the reduction of modeling costs. However, despite their success in the discovery of models from event logs, these algorithms fail in some real world situations. Several information systems don't impose a task flow on the user, allowing him to adapt the execution of a process in a case-by-case manner. If there are no specific guidelines, users may improvise. When these unstructured processes are processed through traditional mining techniques, the result is a *spaghetti process model*. These models are extremely complex, and are of little use for the understanding of the business. Spaghetti models aren't incorrect, but they show that the underlying process is highly unstructured [2]. They may reflect a serious lack of internal organization, and their visual impact can be a great motivation for the inception of process reengineering projects. Several techniques have been proposed to deal with unstructured processes, mainly based on clustering, such as the trace clustering algorithm [3]. They aim to transform a complex problem into a smaller one. The fuzzy miner [2] uses a different approach, with a map abstraction, grouping tasks from a single process model into subprocesses.

Current process mining tools, even when dealing with unstructured processes, assume there is a previously processed event log containing related process instances. The ProM framework [4] is the main existing process mining tool, implementing the state of the art in process mining techniques and covering a vast range of solutions. It works through the input of XML files including instances, events and their attributes, such as task names and originators. External tools facilitate the conversion of data to event logs; however, they don't allow the identification of similar instances.

However, when dealing with information systems supporting unstructured processes, mainly when the user is allowed to input data through free text fields, is it often hard to previously identify which instances should be part of an event log. Even if there is a *subject* field recorded for each instance, it may not be reliable, being too broad, too specific, biased by personal preferences or contain incorrect data. Related instances may be registered with close but not equal information. These nuances are usually not explicit in the database, and knowledge about the correct separation of processes may be hard to obtain, inexistent or hidden. Thus, the discovery of process models should be done in an explorative manner [2]. Extracting an entire database of instances and importing it into the ProM framework, however, is impracticable, and useful data would be lost. Reloading an event log for each mining attempt is also counterproductive.

To deal with this issue, the MANA approach was developed, integrating identification and process mining tasks into the same tool. Instead of importing event log files, it works with a canonical database, containing all process instances extracted from an information system. Instances are selected through sets of filters that define a *process query*. This filtering step allows the user to explore the database and identify relevant processes and tightly related instances, gaining knowledge about the organization. A query can then be processed through existing process mining and analysis techniques.

2 The MANA Approach

The MANA approach was developed to deal with the problem of selecting related process instances when the source information system is not aware of recommended task flows (allowing any task sequence to be recorded) and reasonable process types are not predefined. It is centered in a canonical database containing all instances extracted from an information system, recording data such as subject, description, origin, status, stakeholder, timestamps, tasks executed and their originators. This database allows the exploration of semantically relevant data, enabling users to enhance their knowledge about the processes under analysis. A tool was built to support this approach. It is worth noting that the Aris Process Performance Manager tool [5] also includes process mining functionality using an instance database. However, it assumes that all processes are structured, using process types as the basis for its analysis. The MANA tool was also designed to have a small learning curve for process analysts who are not experts in process mining. The ProM framework, although being the reference in the field, requires a great understanding of its techniques to achieve useful results [1].

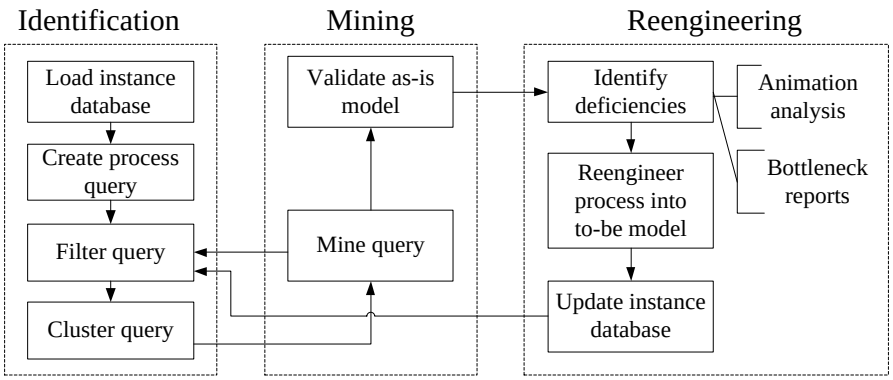

Fig. 1. Workflow supported by the MANA approach

The proposed workflow, shown in figure 1, is split into three phases: *identification*, *mining* and *reengineering*. The *identification* phase involves the selection of related instances. Initially, data from an information system containing process instances and events should be loaded into the canonical database. This can be done, for instance, through ETL (Extract, Transform and Load) tasks. The next step is the creation of a new *process query*. It is defined by a group of filters over the instance database. A query should contain related instances, which can be used for later tasks. While it begins contemplating the entire instance database, this cluster is incrementally narrowed with each added filter. Manual filters, for database exploration, and automatic clustering are supported. The following attributes are currently available for manual filtering: *subject, description, status, stakeholder, year, source, task, first unit, last unit and participating unit*. Filters are selected through searches over the current query's instances. The returned values are sorted by the number of instances that match each result, since the user may be searching for the most relevant processes to model. For example, a search for the value *contract* in the attribute *subject* may return a list of the following subjects: *insurance contract* (298 instances), *acquisition contract* (88 instances), *acq. contract* (10 instances). The user may then choose to commit a filter including or excluding each resulting value from the current query. For example, he may choose to consider all acquisition contracts, resulting in the filters shown in (1). Filters differing only in their value are joined by the disjunction operator (OR), while each group of filters are joined by the conjunction operator (AND), resulting in an expression in conjunctive normal form.

(subject = Acquisition Contract OR subject = Aqu. Contract) (1)

Once a process query is built, it can be used on the *mining* phase. Its goal is the discovery of as-is process models from previously selected. Currently, the tool supports the Heuristics Miner [6], using the ProM framework's implementation. This algorithm was chosen because it deals well with noisy logs. This process model may then be validated and enhanced through meetings with its stakeholders. To support this, the tool integrates a BPMN modeler. This phase also supports the discovery of the flow between units (instead of tasks), since several process control systems em-

phasize more the *where* than the *what* during the recording of a process event. Each unit in the model can later be replaced by the task (or tasks) it executes through careful analysis. Since BPMN's swimlanes (the default notation for process participants) would not satisfy this use case, *organizational unit* nodes were added to the modeler, having the same behavior as task nodes. This approach will be exemplified later by the case study.

The *reengineering* phase allows the identification of deficiencies, which can be optimized in a to-be version of the model. Specifically, visual tools allow an intuitive impact of the current status of the process, allowing an easy identification of bottlenecks. The animation analysis, shown later in this paper, exemplifies this. It was inspired by the animation supported by the fuzzy miner. The animation is built considering the names of the tasks executed by each instance. Each dependency relation is animated. Each instance is shown as a circle, moving from its current place to the next task. In the MANA tool's implementation, if the progress is paused, and an instance is selected, its full data and flow are exhibited. This case-by-case analysis assists in the detection of issues affecting specific instances. Node colors are also coded in a scale between red and green, based on their delays, which enhances the visual analysis.

Besides the process animation, the tool also allows the analysis of task and unit performance through the generation of reports for each process query. For each task, the minimum, average and maximum delay is calculated. Although this approach does not aspire to support a highly complex process analysis, it allows the user to easily obtain insights into the current status of the process. The reporting module also allows the visualization of charts containing all events related to the task under analysis, with scatter plots indicating each task's delay and timelines similar to the approach proposed in [1]. Finally, updated instances can be loaded to the canonical database. Further filtering may be needed to clean the new data. Date filters can be used to compare an outdated process and its current version, ideally including optimizations that correct selected deficiencies. The reevaluation of the source information system is also predicted, but it will not be studied in further details in this paper.

3 Case Study

The following case study exemplifies the use of the developed tool to mine and analyze unstructured processes. The data was extracted from a process control information system from a public organization in Brazil. It handles the registration of processes and their progress between units. Available data includes process subjects, descriptions, stakeholders, origins and status. For each event, the system records a timestamp and its source and destination units. No task names are recorded for the events, so organizational units were used for process discovery. Due to confidentiality concerns, data such as unit names were replaced by letters. The main goal of the study was to analyze important processes from the Information Technology department.

The workflow begins by the creation of a process query. Since the focus is the IT department, instances that went through it or any of its subunits were searched. Subunits are marked by a slash after its parent unit's name. So, all results from the search

IT/% were added to the query, resulting in filters such as the ones exemplified in (2). The % symbol can match any substring.

unit = IT OR unit = IT/A OR unit=IT/B OR unit=IT/C/D (…) (2)

Although predefined process subjects are available, this field is blank for roughly 2/3 of all instances. They are also too generic (e.g. management and operations), reducing their usefulness for process identification tasks. Instance descriptions were used instead. However, since this data is input through free text fields, it should be handled with care. First, by searching all descriptions, the frequent value *disposal of computer equipment* was selected for further analysis. To allow the inclusion of related instance descriptions, the search *disposing%comp%* was used. All returned descriptions were added as filters to the query, totalizing about one thousand instances. The most frequent results include: *disposal of computer equipment*, *disposal of computer material*, *disposal of computer materials.* (sic) and *disposal of computer goods.* Note that the ProM framework would not allow this kind of exploration and filtering.

The next preprocessing step was the removal of rare exceptions, chosen as instances containing units present in less than 1% of all instances. Although these cases are important for exception detection, they would add unnecessary complexities considering the goal of this analysis. This filtering step removed 19 units from the process query, removing only 27 instances from the query. After the data preprocessing, the query was mined using the Heuristics Miner with its default parameters. Since a reasonable process model was achieved, no further clustering was needed. The resulting process model is shown in figure 2. This model can be further enriched through the addition of BPMN gateways and events. Since units were used instead of tasks during the mining phase, further analysis may be needed to deal with units that execute more than one task.

Figure 2 also depicts the animation of the filtered instances over the process model. The color and the slow flow of instances from units E and G provided a visual cue indicating that they should be closely inspected. Through the generation of a report for each unit's delay, it was verified that unit E took the maximum of 476 and the average of 97 days to execute its task. Unit G took the maximum of 1206 days and the average of 108 days. These values are much higher than those from other units participating in the process. The identification of the exact causes for these delays requires a deeper organizational analysis involving the process' stakeholders.

4 Conclusions and Future Work

The modeling of unstructured processes using process mining techniques is a challenging task. This paper presented an explorative approach that allows a process analyst to incrementally identify related process instances and gain knowledge about the organization. A tool was developed to support this through the construction of process queries, filtering instances from a canonical database. It also aims at providing visual tools and an intuitive workflow. Our case study shows that this approach is successful and can be used when there is no clear separation between process types in the source

database. Future work includes the addition of further filtering attributes and techniques. The support for extra process mining and clustering algorithms is also important for advanced users, allowing further improvement and customization of their results. Note that the tool already allows the exportation of queries to XES files if techniques implemented in the ProM framework are needed. Finally, support for a deeper process analysis is planned, through the use data warehousing technology.

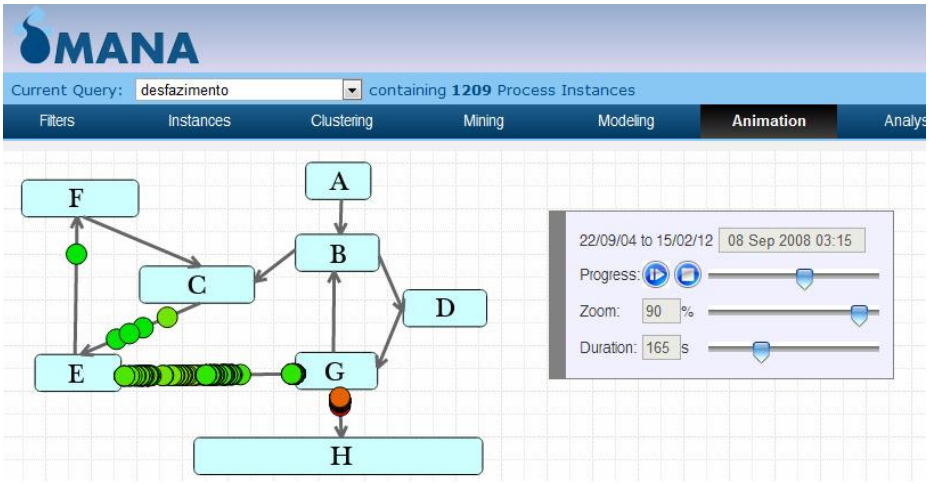

Fig. 2. Animation of the mined process model

References

1. van der Aalst, W.M.P.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer (2011).
2. van der Aalst, W.M.P., Gunther, C.W.: Finding Structure in Unstructured Processes: The Case for Process Mining. Proceedings of the Seventh International Conference on Application of Concurrency to System Design. p. 3–12. IEEE Computer Society, Washington, DC, USA (2007).
3. Song, M., Günther, C.W., Aalst, W.M.P.: Trace Clustering in Process Mining. In: Ardagna, D., Mecella, M., e Yang, J. (orgs.) Business Process Management Workshops. p. 109-120. Springer Berlin Heidelberg, Berlin, Heidelberg (2009).
4. Dongen, B.F., Medeiros, A.K.A., Verbeek, H.M.W., Weijters, A.J.M.M., Aalst, W.M.P.: The ProM Framework: A New Era in Process Mining Tool Support. In: Ciardo, G. e Darondeau, P. (orgs.) Applications and Theory of Petri Nets 2005. p. 444-454. Springer Berlin Heidelberg, Berlin, Heidelberg (2005).
5. Blickle, T., Hess, H.: Automatic Process Discovery with ARIS Process Performance Manager (white paper). Software AG. (2010).
6. Weijters, A.J.M.M., van der Aalst, W.M.P., De Medeiros, A.K.A.: Process Mining with the HeuristicsMiner Algorithm. BETA Working Paper Series, WP 166, Eindhoven University of Technology, Eindhoven. (2006).