Creating Event Logs from Heterogeneous, Unstructured Business Data

Sebastian Pospiech, Robert Mertens, Sven Mielke, Michael Städler and Patrick Söhlke

Abstract Efficient processes give companies the edge required to prevail in global competition. Processes can have a high impact on important factors like product and service quality as well as overall economic efficiency. Hence process improvement plays an increasingly important role in many companies. The first step in process improvement and analysis is understanding the process. While a number of process analysis tools are available, these tools can only analyze processes for which log data (e.g. generated by BPM systems) exists. This paper introduces a tool that allows users to collect and structure traces from undocumented processes like workarounds or improvised processes in order to generate log files. The tool supports query specific ad-hoc exchange of ontologies in order to extract information from unstructured documents containing process traces as well as data extraction components for common databases. It thus bridges the gap between process traces in unstructured, heterogenous documents and process analysis software.

Keywords Process · Mining · Business intelligence · Event log · Transformation · Unstructured data

S. Pospiech (✉)
Cologne Intelligence GmbH, Decision Design, Cologne, Germany
e-mail: sebastian@s-pospiech.de

R. Mertens · S. Mielke · M. Städler
Department of Computer Science, University of Applied Sciences Weserbergland, Hameln, Germany
e-mail: mertens@hsw-hameln.de

S. Mielke
e-mail: mielke@hsw-hameln.de

M. Städler
e-mail: staedler@hsw-hameln.de

P. Söhlke
Next Vision GmbH, Hessisch Oldendorf, Germany
e-mail: ps@nextvision.info

F. Piazolo and M. Felderer (eds.), *Multidimensional Views on Enterprise Information Systems*, Lecture Notes in Information Systems and Organisation 12,
DOI 10.1007/978-3-319-27043-2_7

1 Introduction

In many companies, some processes like workarounds or adaptations to unexpected events do exist without knowledge of the management. They have not been planned and are not documented. Employees do execute certain tasks intuitively as they think it is correct. Often they stick to a certain order and always fulfill a set of tasks in the same manner. They even might have found a very efficient way of execution. However, since there is no documentation, it is almost impossible to define key performance indicators. Hence it is neither possible to measure the efficiency nor to improve such a hidden process. In order to do so, the process with all its facets must be discovered and documented, best in a standardized process model like an EPC. In a highly digitized company, it might be possible to discover such a process by analyzing the data stored in the company's databases, file storages or mail servers. If there is data that is produced by a certain task of process and then used by the next task of that process, the data could serve as a trace to reproduce the processes steps and their connection. In literature there are hardly any methods which are able to fulfill operations which figure out data sets that can describe a whole process. Various approaches range from analyzing transaction logs [1] to building event based data warehouses [2]. While the first type of approaches are simple to implement but are limited to data being stored in one heterogeneous system, more complex approaches are often too difficult to implement, since they need a highly customized solution. One idea is to use ETL-tools like they are used in classical BI in order to receive data from a database that belongs to a process. This data can then be used by the ETL task to receive data from another database. Such a job would represent a process consisting of two tasks which produce data in the particular database. However, ETL tools are designed for huge data sets. They often lack the abilities to make investigations on single data rows and do only work well on structured data like databases. Our approach describes a similar way to discover processes but with the difference that it is designed to investigate a single process execution's trace by simply searching for and analyzing the data it created and used. In a second step, our software provides methods to expand the specific queries in a way that other executions of the process are found and checked for conformance automatically. The final output of our software is an XML-document containing information about the investigated process executions, extended by semantic information describing the process. It can be shown that this structure is functionally equivalent to an event log as it is used by Process Discovery algorithms.

The remainder of the paper is organized as follows: In the next section we discuss related work. Then we will show the limitations of an ETL-based approach and introduce our software and how it works. We will proof the equality of our softwares output compared to an event log and will demonstrate how to transform that output into a classic event log that can be read by Process Discovery algorithms. This step of transformation is important since it connects a multi-source data

analysis of both structured and unstructured data as it is done with our software to classic Process Discovery as it is used in process mining. The paper will conclude with the discussions of the system's role in process analysis.

2 User Story

Imagine Mr. Miller, a business consultant hired by a manufacturing company that has problems with their complaint handling at a specific problem. In order to find out what is going wrong, Mr. Miller has to know what the process of handling customer complaints looks like. Since there is no documentation and no one is aware of a real process but is simply doing his or her job to the best of knowledge, Mr. Miller has to discover the process that probably is behind the complaint handling.

He starts questioning the members of the hotline team. Those note down complaints in the CRM system of the company and link it with the particular customer. Mr Miller can proof that by querying the CRM database and looking for complaints that he takes as an example. For each of these complaints, a serial number of the broken product is attached. He decides to use that serial numbers and looks into the database of the goods receiving department, to check if the broken product has been send back to the company. Thus, the serial number serves as a business key and links the data from one source to another.

Mr. Miller now has reconstructed two steps of the customer complaint handling process: Registering the complaint and receiving and inventory the broken product. This way he can reconstruct the whole process step by step using questionnaires and example data. When he is finished he might want to test if all complaints follow the same process. So he has to reconstruct all the steps for several executions of the process, starting with the entries in the CRM database. He might also want to verify steps of the process that generate unstructured data. A step to inform the customer for example could produce a letter that is stored on the file system of the company. Unstructured data is usually quite more challenging. For efficiently searching in unstructured data it might be necessary not only to search for a certain word but also to look out for synonyms or hyponyms in a certain context. Especially when verifying the reconstructed process with other executions, it might be unsatisfying to always look out manually for corresponding unstructured data.

3 Related Work

Our work is settled in the field of Process Mining that is a part of Business Process Management. One task of Process Mining, Process Discovery, deals with the creation of Process Models for describing and documenting business processes. Creating Process Models can either be performed manually or by using Process

Discovery algorithms [1, 3–7]. These algorithms take event logs as an input and process them to create the model. PROM is an Open Source collection of Process Mining tools that provides various Process Discovery algorithms. Our work focuses on generating event logs that can be loaded into and processed by the PROM software [8].

In this paper we address the problem of creating the event log to be used by a certain discovery algorithm. Usually this log data is being gained directly or indirectly from digital information systems that are used by an agent who performs an undiscovered process [9]. Since business companies often have heterogeneous structures, especially those which have grown very fast or through corporate merges, it is difficult to extract this data. It might be distributed over various databases or simply be stored in unstructured formats on file- or mail-servers. One idea is to use log-files that are originally created for debugging and maintenance purposes: In [6] the logs of a subversion system are used to create an event log describing the software development process which a development team uses to implement their software. A second approach is simply to use the features of a workflow management system to gain the log [10]. A third idea is to use an event-data warehouse. Such a warehouse is structured like a classic warehouse used for Business Intelligence, but it tracks event data for example generated through the daily work of a mail-order company with their mail tracking system. Those three approaches are quite good for a certain purpose, but they all do lack in different things. Most approaches described in literature do only work in a homogeneous environment and only with structured data. They are not capable of creating logs that contain data from various systems. Business Processes which affect several databases can't be recognized completely by them.

4 Why not Use ETL?

The usage of ETL tools to track events, by building up an event data warehouse (see [11]) or in smaller environments simply a data mart or csv output does not have the disadvantage of being limited to homogeneous environments. One of the main purpose of ETL is to connect data from various, heterogeneous systems and integrate it into a specified output.

Other than transporting data for Business Intelligence needs, which almost solely derives from structured sources like databases, business processes often generate unstructured data like e-mails or word files. ETL tools are limited to structured data by design. State of the art ETL software like Talend Data Integration or Microsoft SSIS are not even able to read from an unstructured source. So it is hardly possible to find information or a trace of a hidden process in an unstructured document, using ETL.

In order to run Process Discovery on a hidden process, it is necessary to use the data created by that process as validation. If there is a step in the process in which a letter is send to a customer, than there must exist a document for that letter

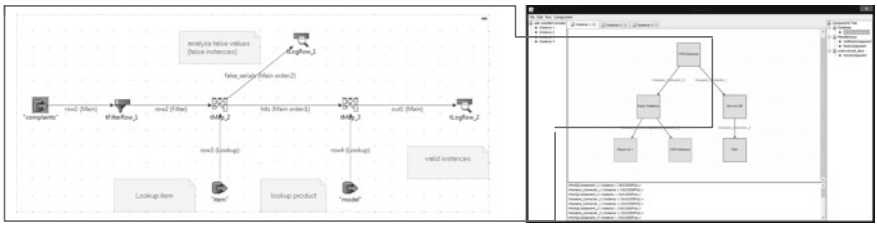

Fig. 1 Comparison between a talend open studio process (*left*) and our Process Explorer

somewhere in the company's network. If it is also possible to find ten additional letters that do all look similar (e.g. having the same keywords like 'claim' or 'return'), then the process has probably been executed ten times and in ten different contexts with that step. In order to use this information in an event log, the analyst trying to discover the process has to analyze the data, try to find such a letter, identify key words and then validate if there exist similar letters to identify other examples of hidden process. For each of those letters, the key information has to be extracted in order to serve as an identification for that instance and to help identify further steps of that process. ETL tools can hardly support users in this task. Their abilities to work with single data sets, visualize and analyze them, are limited and not very comfortable. ETL environments have options to view data just for debugging purposes not to work with the data as a key element to fulfill the task of data extraction.

On the left side of Fig. 1 the ETL tool Talend Open Studio has been used to validate the first two steps of the example use case. The job looks technical, hard to understand for someone who is not a developer. Queries to unstructured components can not be covered and the result of the job shown in the example does not generate an event log but simply prints out the found data. In order to use it for Process Discovery additional transformations which make the job even more complex would be necessary.

5 Process Explorer

Our idea was to design and create a tool that is capable of supporting a consultant or a business analyst in the Process Discovery task as described in the example use case. Our tool is designed similar to classic ETL tools on purpose. The basic idea of retrieving data from various data structures is the same and our target group is familiar with these tools.

On the right side of Fig. 1 a screenshot of the application is shown. The two steps in the red area are semantically equal to the whole Talend ETL process on the left side. The process is being designed in a graph based user interface. Components can be configured to retrieve data from the source systems. A connector is a

directed edge that copies data from a source component to a target component. In the example use case, the fist component could access the CRM system and get the serial numbers of the products for which complaints are open. The connector copies these serial numbers to a second component, which then can use the serial number as a business key to look for corresponding information either in another database or on a file server. If there is exactly one corresponding result, this should be the one which was generated in that specific process execution of that complaint. If there is no result or even more than one, this could be a hint on a mistake during the process execution (assuming that the query used to search in the data source is correct). In this case our tool highlights the component in which the query returns an unsatisfying result and the user has to analyze the data for that specific execution.

Because of the need to make detail analysis of certain process executions and its corresponding data, our tool does not behave like an ETL tool and processes many different data sets at a time. It only processes one process execution at a time—each execution in an individual tab. This makes it much easier for the user to analyze how a specific execution behaves and if it is a good representative of the process to be discovered or if it is incorrect. The graphs' structure is not as technical as the one in an ETL tool.

5.1 Searching in Unstructured Documents

Furthermore, our tool can also find information in unstructured documents. In our prototype we provide a component that is capable of handling all basic document types like doc, docx, rtf or odt. Information items stored in unstructured files are neither that easily processable like structured ones, nor is a search in unstructured documents that easy, since natural language is being used.

In order to address the problem of searching information we used a full text search based on elasticsearch and provided different kinds of search methods. First of all, simply searching for a a set of words. Second, searching for a regex pattern and last but not least, searching for a word and all of it's synonyms.

In order to be able to search for a synonym we developed a search engine, that combines the elasticsearch with a triple store to create ad-hoc ontologies for a certain context. We could for example create an ontology describing the semantics of a company's structure. This enables our search component to answer queries like 'Find all documents on the company's file server, which are signed by someone that works in the same department like Mr. Miller'. This feature is important for solving problems when searching in a domain where natural language is used and when trying to find different executions of processes automatically. A business analyst might know because of his examinations that a certain document has been generated by Mr. Miller in the complaint case number 4711, but he does not know if documents of the other complaint cases he wants to use for process re-engineering have also been written by Mr. Miller. Probably they haven't been. Using such an

ad-hoc ontology enables a business analyst to search for all members of a business domain without knowledge of all its members. Using predefined ontologies for the English language could help finding documents in which the terminology is not well-defined and may vary from author to author.

6 Connection to Process Discovery

In literature [9, 12], the field of Process Discovery finishes with the creation of a process model. The graph designed in our application so far is not a standardized process model like an EPC or a petri-net. It is rather a trace of the process which has been uncovered by the data (the data is a footprint in that metaphor). What our tool does is actually a step before Process Discovery as being illustrated in Fig. 2. It takes the trace of a process, adds semantic information like a description of the activity and creates an event log as an output, which can serve as an input for a Process Discovery algorithm.

The proof that the graph designed by our tool is compatible to an event log has been introduced in detail in our recent work, see citation number [13]. What the user actually does is creating an event log that is simply designed as a graph and the generalization to all executions. Therefore one basic element of our tool is to convert an event log into a format that is compatible with the PROM framework in order to apply it on a Process Discovery algorithm. The export feature simply iterates through the graph and creates a csv output. For each process execution stored in the graph an ordered list is being generated. Each row in this list represents one event, with the name of the activity, an optional description and an optional

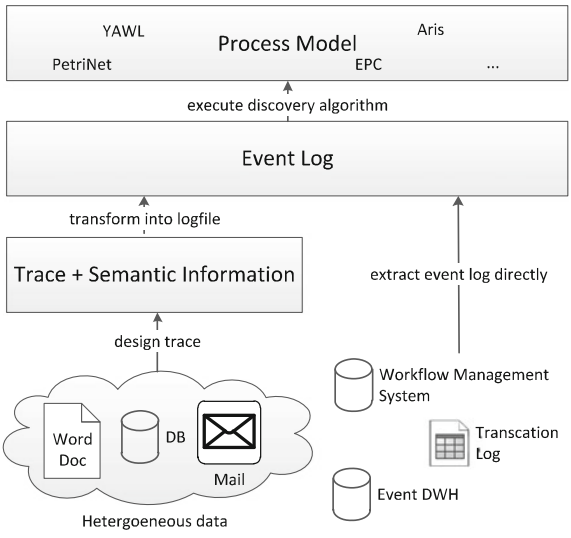

Fig. 2 Process Discovery using our approach (*left*) verses the classic approach

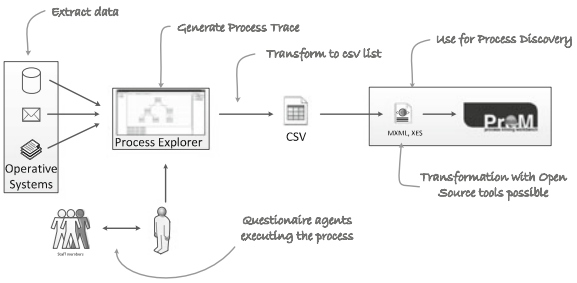

Fig. 3 Sample process designed using the Process Explorer

timestamp. The whole workflow of reengineering a process using Process Discovery (e.g. using Nitro and PROM) and our Process Explorer to create a process trace representing an event log, can be seen in Fig. 3: A business analyst uses our tool while talking to the people executing the process to create a trace out of the data provided by operative systems. He exports the trace into a csv list, which then can be used by state of the art process mining/Process Discovery software.

7 Conclusion and Outlook

In this paper we introduced a software to be used as a tool supporting process reengineering by using heterogeneous business data. We have shown that the tool plays a role in Process Mining and its output can serve as input for Process Discovery algorithms. In a follow up project the implemented prototype has to be improved and tested using more advanced ontologies for querying unstructured documents in a complex business environment. Beside domain specific ontologies based on company specific structures, ontologies used to improve querying natural languages like synonym registers could be integrated and used.

References

1. Cook, J.E., Wolf, A.L.: Discovering models of software processes from event-based data. ACM Trans. Softw. Eng. Methodol. **7**(3), 215–249. ISSN:1049-331X (1998)
2. Schiefer, J., et al.: Event data warehousing for complex event processing. In: Loucopoulos, P., Cavarero, J.L. (eds.) IEEE, RCIS, pp. 203–212 (2010)
3. van der Aalst, W., van Hee, K.M.: Workow Management: Models, Methods, and Systems. Cooperative Information Systems. MIT Press, Cambridge (2004)
4. Agrawal, R., Srikant, R.: Mining sequential patterns. In: Proceedings Data Engineering 95, ICDE '95. IEEE Computer Society, Washington, DC, USA, pp. 3–14 (1995)
5. Das, S., Mozer, M.: A unified gradient-descent/clustering architecture for finite state machine induction. In: Cowan, J.D., Tesauro, G., Alspector, J. (eds.) Morgan Kaufmann, NIPS, pp. 19–26 (2003) ISBN:1-55860-322-0

6. Duan, B., Shen, B.: Software process discovery using link analysis. In: IEEE 3rd International Conference on Communication Software and Networks (ICCSN), pp. 60–63 (2011)
7. Werf, J.M., et al.: Process discovery using integer linear programming. In: Proceedings of Petri Nets 08, pp. 368–387. Springer, Heidelberg
8. van Dongen, B.F., et al.: The prom framework: a new era in process mining tool support. In: Proceedings of Petri Nets 05, pp. 444–454. Springer, Miami (2005)
9. van der Aalst, W.M.P.: Process Mining: Discovery, Conformance and Enhancement of Business Processes, 1st edn. Springer, New York (2011). ISBN:3642193447, 9783642193446
10. van der Aalst, W., Weijters, T., Maruster, L.: Workow mining: discovering process models from event logs. IEEE Trans. Knowl. Data Eng. **16**(9), 1128–1142. ISSN:1041-4347 (2004)
11. Casati, F., et al.: A generic solution for warehousing business process data. In: Proceedings of Very Large Data Bases 07. Vienna, Austria: VLDB Endowment, pp. 1128–1137 (2007)
12. van der Aalst, W.M.P., et al.: Process mining manifesto. In: Business Process Management Workshops '11, pp. 169–194 (2011)
13. Pospiech, S., et al.: Exploration and analysis of undocumented processes using heterogeneous and unstructured business data. In: Proceedings of IEEE ICSC 2014, pp. 191–198 (2014)