End-to-End Process Extraction in Process Unaware Systems

Sukriti Goel[1], Jyoti M. Bhat[2], Barbara Weber[3]

[1] BPM Research Group, Infosys Labs, Infosys Limited, India
Sukriti_goel@infosys.com,
[2]Information Systems, Indian Institute of Management, Bangalore, India
Jyoti.bhat@iimb.ernet.in
[3]Department of Computer Science, University of Innsbruck, Innsbruck, Austria
Barbara.Weber@uibk.ac.at

Abstract. Knowledge of current business processes is a critical requirement for organizational initiatives like compliance management, regulatory reporting, process optimization, reengineering the IT systems and outsourcing. Existing process discovery techniques expect process execution information or event logs while organization's business processes are often executed on heterogeneous systems across different departments, by integration and data hand-offs between systems. Traditional information systems, however, are designed for storing and processing transaction data which persists in databases and other data storage mechanisms. In this paper we identify the challenges and propose a solution for extracting end-to-end processes from persistent process execution data available in multiple heterogeneous applications. The approach consists of analyzing persistent system data to identify and obtain events in a non-intrusive manner. The approach to get the end-to-end process involves a combination of data and process mining.

1 Introduction

Global customers and multi-location presence of enterprises implies that organizational processes are executed leveraging technologies like the internet, web services and cloud computing, yet safeguarding the existing investments in legacy applications, enterprise packages and products. The business process logic and rules are coded in algorithms, batch jobs, database constraints, etc.; and the process flow is not explicit as it is achieved by integration of multiple applications. A single end-to-end business process is implemented across several applications many of them being process unaware systems [1]. Over time applications are modified due to business requirements of various stakeholders, technology enhancements and regular system maintenance. But existing process documentation is usually not updated with the systems' changes. As a consequence the process documentation and the business process supported by the organization's information systems (ISs) run out of synch. Process stakeholders, however, need to know the As-Is processes executed in the systems for controlling and improving business process performance [19]. In addition to the process flow, varying level of process details are needed, For e.g., average cycle time of execution, process measures and SLAs based on the nature of the business process initiative.

Common approaches to discover or extract business processes like process elicitation by interviewing different stakeholders, code study and study of existing system documents and manuals can be used to elicit the process flows. However, these approaches are insufficient to obtain other details like process performance measures, cycle times and queue times. Moreover, they are costly, time consuming, and heavily dependent on business knowledge of the stakeholders [2]. Process mining addresses this problem by extracting the business process from the information systems supporting them at an operational level. Process mining uses event logs where each event refers to a case and is totally ordered [6]. While multiple automated process mining tools [3] are available, they expect event logs provided in a certain format as available in a process aware information system (PAIS). In traditional information systems event logs are not available since logging mechanisms are typically missing and the processes often execute across multiple applications. To address this problem of unavailability of event logs in traditional information systems, static analysis of code followed by instrumentation and dynamic analysis has been used to generate the log files [4, 20, 21]. However, making modifications to production systems is risky and involves many stakeholders. Moreover, many organizations do not allow modifications to the existing systems in any form.

We propose a process extraction method which can extract multiple end-to-end processes executed in various heterogeneous systems in a non-intrusive manner. In this paper we will list the challenges for end-to-end process extraction from the process execution data of process unaware information systems (cf. Sect. 2), provide an overview of related work (cf. Sect. 3). We will present the proposed method for extracting the end-to-end process by tracking the process execution data (cf. Sect. 4) along with a case study (cf. Sect. 5). Finally, we provide conclusions and discuss future work (cf. Sect. 6).

2 Process Extraction: Challenges

C1 - Missing process awareness: The concept of a business activity is missing in traditional ISs as the process flow is implicit and implemented by invoking methods, procedures and programs (callable units). This creates a challenge of not knowing what business activities are executed [20, 21]. PAISs have explicit process definitions and execute the process as sequences of activities and maintain the status of execution [8]. Often traditional ISs are viewed as a graph where nodes are the different callable units and propose an approach to map each callable unit to a business activity for process mining

[20, 21]. Using this information for discovering a process requires instrumentation of code.

C2 - Systems Landscape: As an end-to-end business process may span multiple applications and technologies, the challenge is to identify all the systems involved in the process. For example, the order to remittance process of a software service provider spans across 30 applications like CRM, project allocation and delivery, time sheet management, and finance applications.

C3 - Granularity: The difference in granularity of business activities and data traces pose another challenge as it is difficult to decide which data traces are to be considered for processing. Business activity 'interest calculation' is of interest to process stakeholders, but the details of how interest is calculated (e.g., 'get interest rate', 'get account details') are too fine-grained to be included in a process model. Additionally, different data storage systems record execution data at different levels of granularity. The data traces in log files and audit logs are typically fine-grained (e.g., event for entering data for one particular attribute), while one entry in a transaction table in DBMS might be related to multiple activities which occurred on the business entity.

C4 - Unstructured data: Even in automated business processes, certain parts of the process information may be in an unstructured form like scanned documents or e-mails. It is difficult to interpret and associate unstructured data with the business process and its activities. It is an even bigger challenge to interpret data from scanned documents without indexing information and emails written in natural language.

C5 - Multiple process identifiers: Traditional ISs do not have a unique identifier for a process instance. The multiple entities involved in process execution are stored with different identifiers in the process execution data and hence there may not be a single identifier that flows across the systems which can be used to identify an individual case [11]. For example, in an order management system, the entities involved are *order*, *item*, *invoice* and *shipment,* thus, multiple identifiers such as *order id*, *item id*, *invoice id* and *shipment id* exist. The challenge is to correlate the various entities and the conditions on which correlation can be done.

C6 - Overwritten data: Depending on the design of the data structures, data gets overwritten in some transaction data sources such as DBMS and indexed files. For example, the status of an order may be maintained using a *status* field in the *order_details* table which gets updated during system execution with values 'created', 'invoice created' 'payment received' or 'shipment sent'. In some cases the time stamp of the event gets overwritten each time a change is made in the tuple (e.g., column "last_modification_date' is updated each time a change occurs in the row and the modification date of previous events is lost).

C7 - Missing timestamps: Process extraction from persistent data has another challenge related to timestamps. Typically, data traces are logged at the completion of an event, thus, the timestamps available are usually completion times. The start times of the activities are usually not available. In addition, there may be some activities which are recorded in the persistent data sources with no information of date of execution. Such actions may be listed in the process model but order of execution cannot be known.

3 Related Work

Process discovery and documentation is of interest for many organizational initiatives like process improvements, systems reengineering, process performance monitoring, system maintenance and outsourcing. Various approaches and tools are used to understand and document the business process executed in ISs. Process elicitation by interviewing business users and conducting workshops with the process stakeholders is a popular method for process discovery. The drawback of process elicitation is its dependence on business users who narrate the process and the consultants who understand and document the process. In addition to being costly and time consuming, this method depends on the process knowledge with the people and is vulnerable to human interpretations and political influences [2].

There are other manual techniques where the task execution data is collected while users are working on the tasks. These methods expect either the process user or the external observer to note execution data like instance id, task name, task start and end times for each of the task while working on it [16, 17]. These methods reduce the overhead on the business user, but introduce extra cost by demanding extra work from process workers or by introducing the role of observers. Though time consuming, this method is usually used for analyzing processes in the context of process improvement initiatives (e.g., six-sigma).

Business provenance technology [9] helps identify end-to-end business processes, especially unstructured processes, across heterogeneous systems by collecting, co-relating and analyzing operational data. This approach involves creating a provenance graph for the specific application needs like compliance monitoring. The provenance data generated depends on the information about the business operations that is to be extracted from the applications. Moreover, the provenance data is generated by accessing application events from event reporting middleware or by processing the application data and identifying the events. This approach expects the applications to generate the provenance data in a pre-defined format; otherwise the applications need to be instrumented.

Process Mining has been around for more than a decade [10]. Existing process mining techniques assume the presence of event logs [12], which is however not always that easy. Though process mining techniques have the drawback of not being able to discover the manual part of the process, they are gaining popularity for the processes which are automated.

ProMImport [8] is a framework for converting logs in any format into event logs as required by many of the process mining techniques. Though the framework is extensible to add more input formats, it does not provide suggestions on generating the event logs for processes executed in process unaware systems and processes executed across multiple heterogeneous systems.

To handle the challenge of generating event logs for traditional ISs and process unaware systems with instrumentation a combination of static and dynamic analysis of source code has been proposed [20, 21]. The code needs to be modified in the production environment to generate the event logs while the process is executed. Hence, for long running processes, the process would need to be executed for a long time to get enough information (i.e., certain processes have a periodicity of months, with annual events and seasonal differences).

Process spaceship [11] is a tool for semi-automatic definition of a process space over heterogeneous IT systems in an enterprise. In particular, it allows analyzing the information items in an enterprise, correlating them into process instances and generating process views. This work addresses process discovery in heterogeneous systems using multiple data sources, though it does not specifically refer to IT systems. The approach is based on systems using web services with message exchanges between them. Moreover, the process spaceship approach does not deal with heterogeneity of the enterprise events and data sources, but is dependent on Extract, Transform and Load (ETL) tools. Another observation is that this work considers each tuple as one event which may not be true. In many practical settings, a tuple may contain information of multiple events

Existing Business Activity Monitoring (BAM) tools like ARIS Process Performance Manager (PPM) [4] can reconstruct the execution of each transaction from start to finish by combining the process-relevant data from multiple IT systems (e.g., ERP, workflow, legacy systems, etc.). Organizations can obtain a comprehensive performance overview of their business workflows using process indicators and a graphical visualization of the actual structure of their transactions. ARIS PPM collects information related to the transactions during execution and hence does not have to correlate the transactions. But to discover the process, sufficient number of transactions needs to be captured by ARIS PPM over a period of time. This poses a problem specifically for long running process.

Other methods have been proposed for mining business processes from event logs when there are missing case ids [22, 23]. But these approaches do not handle traditional IS where the process execution data resides across multiple heterogeneous systems. The method of process extraction described in this paper complements to existing methods of process mining.

4 Process Extraction (PE) Approach Overview

As multiple robust process mining algorithms are already available [6, 11-16], the proposed method concentrates on the generation of event logs in a non-intrusive manner using persistent process execution data, for e.g. transaction data. The approach does not intrude the system either by adding code or introducing probes. Additionally, as historical data in systems are archived due to business and regulatory requirements, generating a sufficiently large event log to address long running processes is not a problem. The event logs can be then used to extract the process flow and apply business intelligence techniques [19] to generate the process models. Figure 1 provides an overview of the proposed technique.

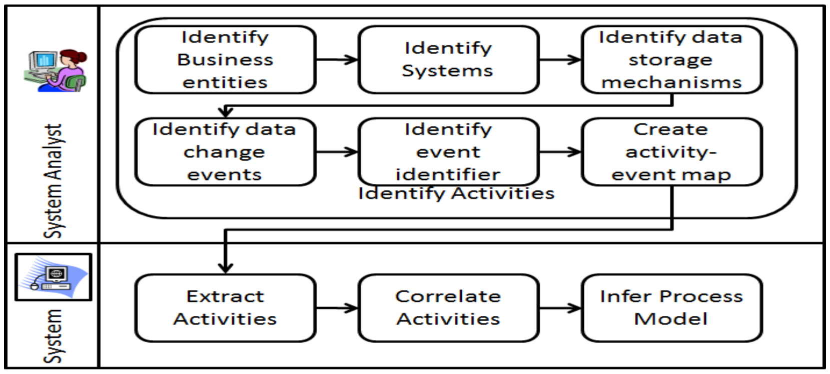
System Analyst
Identify Business entities
Identify Systems
Identify data storage mechanisms
Identify data change events
Identify event identifier
Create activity-event map
Identify Activities
System
Extract Activities
Correlate Activities
Infer Process Model

Figure 1: Process discovery approach

A. Identify Activities

The first step in order to identify activities is to know the data relevant to the end-to-end process. The information about the business entities and systems involved in the process is acquired from business users and IT staff respectively. Once the systems are identified, all data storage mechanisms, where the process execution data is stored are identified such as DBMS, system log files, middleware log files or flat files. Then we analyze the data sources and identify the data change events related to *create* and *update* operations in the data sources. It may not be possible to identify some *update* operations and any of the *read* and *delete* operations from the data sources in the absence of audit logs. Typical data traces left by process execution are - insert row in a DB table, update a row in DB table, create new file, add a new log entry in a file, send a mail or receive a mail. Each of these data operations corresponds to at least one event. We ignore the data operation performed on master tables and any data operation with no time stamp.

The granularity of identified data events varies from one data source to another. Coarse grained data events from transaction database tables may be clubbed with data events from other data sources, e.g., web logs or application log files. Also, if the identified events are too fine grained then some of the events can be ignored either while mapping them to activities or during the process model inference using mining algorithms.

Next, the correlation identifiers for each data event are marked. Applications store entity id and time stamps in one form or another to distinguish the instances from each other for future processing [7]. The applications store this information in various forms like transaction tables in database, audit logs or flat files. Usually, the correlation identifier is the tuple identifier which includes composite identifiers. Data change events in DBMS and indexed files can be represented by 1) the table name, 2) an id (i.e., correlation id), 3) a condition which helps mapping multiple events from the same tuple to different business activities and 4) a date (i.e., timestamp). Table 1 provides several examples. Table *invoice_details* has both order_id and invoice_id columns which help in correlating events captured from two different table. Presence of foreign keys in the database schema makes it easy to correlate events read from different database tables. Each of the events is mapped to an activity as mentioned in the last column.

Table 1: Representing Events and Activity Mapping

Table	Id	Condition	Date	Activity
order_details	*order_id*	*null*	*create_date*	*Create Order*
order_details	*order_id*	*Status='cancel'*	*cancellation_date*	*Cancel Order*
order_details	*order_id*	*cancellation_date !=null*	*cancellation_date*	*Cancel Order*
invoice_details	*order_id+ invoice_id*	*null*	*create_date*	*Create Invoice*

B. *Extract and Correlate Activities*

Once the activities are identified, the data sources are parsed and the events extracted in an automated manner. Each event is mapped to a business activity based on the mapping rules defined in the previous step. This creates a cloud of unrelated activities.

As the objective of correlating the activities is to generate an event log comprising of process instances of the business process, the events are sorted by timestamp. The events are correlated using composite disjunctive conditions [11]. There is no prior knowledge of relations between the events as conversations and time constraints do not exist in the data sources considered. The process instance identifier or correlation identifier is progressively built using the event identifiers based on the events which match the existing correlation identifier. If correlation identifier of the activity matches with any of the correlation identifier of the existing process instances, the activity is added to that instance. If the activity has more than one identifier (i.e., a composite identifier), all the correlation identifiers are added to the process instance identifier. If the activity identifier does not match with any of the existing process instances, a new process instance is created and the correlation identifier of the process instance is set as the event identifier of the activity. In some cases, one activity can correlate with multiple activities of same type, e.g. *create order* activity can correlate with multiple *create invoice* activities as one order can have multiple invoices created.

We explain our approach for activity correlation using an example process with two entities: order and invoice. The time sorted activities of the process are shown in table 2.

Table 2: Activity Cloud

Activity	Id	Time stamp
create order	*order_id = 1*	*t1*
create invoice	*order_id=1,invoice_id=100*	*t2*
create order	*order_id=2*	*t3*
receive payment	*invoice id=100*	*t4*

When 'create order' is processed by the correlation algorithm, there is no process instance with correlation identifier "order_id=1", thus, a new process instance is created with correlation identifier “order_id=1”. When activity ‘create invoice’ with identifier “order_id=1, invoice_id=100” is considered, it matches the process instance with identifier “order_id=1”. The correlation identifier of that process instance is updated to “order_id=1, invoice_id=100”. Again ‘create order’ with order_id=2 does not match any existing process instance so a new one is created with correlation identifier “order_id=2”. When activity ‘received payment’ with identifier “invoice_id=100” is encountered, the event identifier matches with the process instance identifier “order_id=1, invoice_id=100” (using disjunctive correlation conditions). The event is added to the process instance, but the process instance identifier is not updated.

The process instances can be built from the extracted activities in the event cloud in an automated manner.

C. *Infer Process Model*

The process instances obtained by correlating the activities are used to obtain the process model. Most of the existing process mining algorithms can be applied to obtain the end-to-end process model [3, 6, 12]. The process model is refined using various methods available in process mining tool, which we refer to as discovered process. The discovered process is reviewed with the process stakeholders and the final process signed off after making the required changes in the process. Rozinat et. al. [5] define fitness and appropriateness metrics to compare the discovered process and the existing process model documented. But here we use the discovered process and extend it to create the final process model, as existing process models and documents are absent. The metrics, fitness and behavioral appropriateness are not relevant here, as the process model is discovered using execution logs. Rather the process stakeholders may want to add paths to the process model which were not discovered using the given method. In such cases, we recommend to using persistent data of a longer period to actually validate the additions made by process stakeholders. There may be some undesired activities in the process model which can be removed by the stakeholders at the time of review. Currently we manually convert the process model obtained in petri net or heuristic net into BPMN. Hence structural issues are handled manually and hence assessing the structural appropriateness of discovered model is not of concern.

There is a need to measure completeness of discovered process model as compared to final process model. Completeness can be measured for activities and transitions. For calculating completeness of activities C_A and completeness of transitions C_T we use following:

$$C_A = \frac{N_d}{N}$$

$$C_T = \frac{T_d}{T}$$

where:

N_d	is the number of activities in discovered process
N	is the number of activities in final process
T_d	is the number of transitions in discovered process

T is the number of transitions in final process

We can get a weighted average of C_A and C_T to arrive at single measure of completeness of process as follows:

$$C = 0.5\ C_A + 0.5\ C_T$$

5 Case study: Purchase Order Process

We applied the process extraction approach to discover a real-life end-to-end process spanning three heterogeneous systems. In the case study we aimed at the discovery of the purchase order process from the process execution data of a large services organization. Goal of the case study was to evaluate our technique by measuring the completeness of the discovered process vis-à-vis the final process signed off by the process stakeholders.

Obtaining Event Logs

With inputs of process stakeholders and system owners, we identified that the purchase order process executes across two departments (i.e., purchase and finance department) using the following systems and data sources:

1. Purchase Request (PR) System - used by the employees of the organization, uses a RDBMS and partly mail based for approvals
2. Purchase Order (PO) System – used by the back office of the organization, uses a RDBMS for storing data
3. Accounts receivable (AR) system – uses RDBMS for storing data with detailed audit logs

We found that when a new request is created, a new row is inserted in the *Purchase_Requisition* table in the PR system. When a new purchase order is created, a new row is inserted in the *Purchase_Order* table in the PO system. For obtaining senior management approval, a mail is sent to the approver in controlled natural language and the approver replies to the mail with approve or reject along with comments, leading to challenge C4.

When identifying the *insert* and *update* operations in the data sources, we detected that some important data gets overwritten. The *status* column of table *Purchase_Requisition* gets overwritten each time the status of a request is changed (i.e., the value changes from *created*, *PO generated*, to *approved* to *completed or cancelled*) as discussed in challenge C6. In some cases the data gets overwritten in one table, but at the same time a new data record is created in another table. For example, when the *status* field is updated to *PO generated*, a new row is inserted into *Purchase_order* table in PO system and is captured as *create purchase order* activity. Therefore, the challenge of overwriting data could be overcome in our case to some extent by considering multiple systems and data sources.

In our case study, we found the same data change events available both in transaction tables and audit logs. Audit logs and audit tables due to their fundamental purpose, capture all events of importance in the system for reporting, compliance monitoring and future analysis and data is not overwritten or deleted. Whereas in transaction tables, data is sometimes overwritten and the tables only store information that is needed for further processing of the process instance. In the AR system, for the same process we identified 24 distinct data events using the audit logs and 10 events with transaction tables leading to different granularity of extracted process as discussed in challenge C3. Using audit logs, more fine grained process can be obtained as compared to using transactions tables. One important finding of our case study is that audit logs should be used wherever possible to identify data events.

We used the keyword search on the email sent and received by the PR system as controlled natural language is used. We could identify the name of the approver, date of approval requested and action taken by the approver along with the date.

We found the unique identifiers of the tuples were sufficient to identify the events. We used the primary keys of the tables (e.g., *purchase_request_id*, *purchase_order_id*) as correlation ids. As approval request and response mails had the *purchase_request_id*, the same could be used to correlate the events across email and PR system. Rules for mapping individual data events to business activities were defined. The events were then extracted by reading the data sources and correlated to generate the event logs or process instances in a fully automated manner using the correlation algorithm.

Discovering the End-to-End Process

We used the Heuristic miner plug-in of ProM to discover the end-to-end processes from the event log generated. Our event log was for a 6 months period and it contained 6700 instances and 51200 events. The maximum number of events per instance was 26, while the minimum was 1. The extracted process was then reviewed with the stakeholders and system owners to evaluate the completeness and correctness of the process.

We measured the completeness by comparing the process as signed off by process stakeholders with the process discovered by ProM. We could discover the process with a completeness CA of 0.75 and CT of 0.65. The process completeness was not 100% as some of the process execution data was lost as the data was overwritten in the transaction tables with no corresponding entries in audit logs or other data sources.

It took us a manual effort of 3 days to go through the systems, identify the data sources, tables, unique identifiers, etc. Manual effort was also spent in converting the heuristic-net model into a BPMN model, which was around 2 days.

Based on our experience of applying the proposed method to extract multiple different processes, we find that the results depend on the availability of the persistent data. We could achieve completeness of 50% to 95% based on the data available. In presence of audit logs or multiple sources of inputs for the same system, e.g. database transactions, system logs, etc., the completeness is much higher as compared to using transactional data only. The completeness also depends on whether the transaction logs contain all possible paths which the process can take like exception handling and infrequent events. This to some extent is handled by collecting the transaction logs for a significantly large period.

6 Conclusions and Further Work

Automation-assisted process extraction holds a lot of promise today. It offers the capability of discovering the process as executed in non-process aware systems. Automated process extraction does not interfere with the execution of processes as it uses available historical data to discover the process. The process is extracted from persistent process execution data available in various data sources. Business transaction data is usually logged during execution for further processing. A limitation of our current approach is that business activities that do not leave a trace in the persistent data cannot be extracted; manual activities and the processing done in system/program memory fall under this category. Transaction data getting overwritten can be handled to a certain extent by identifying another data source of the system where the data event may persist.

The challenges related to identifying the system landscape and granularity of the process can be resolved with the help of domain experts. The proposed approach takes care of multiple process identifiers of the process across systems by using the disjunctive correlation method. Though we have not used text mining on the emails in the case study, there exist techniques to identify data events from unstructured data [9], which can be applied to further improve the effectiveness of our method.

The availability of process information required for Business Process Intelligence (BPI) and process optimization depends on the quality of data traces. Usually, the data traces do not store data at the start of an activity, so it may be difficult to generate information like average time to complete an activity. Also the currently available mining algorithms do not mine the process flow rules along with the extracted process.

Future research can explore mining the business flow rules along with process discovery using the data traces available in the systems. Another area of work is to improve the quality of event logs to generate enough process information as required by process optimization and BPR initiative.

7 References

[1] F. Leymann, W. Reisig, S. R. Thatte, and W. M. P. van der Aalst, “The Role of Business Processes in Service Oriented Architectures”, number 6291, Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, (July 2006)

[2] L. Verner, “The Challenge of Process Discovery”, BP Trends; May 2004

[3] B. van Dongen, A.K. Alves de Medeiros, H.M.W. Verbeek, A.J.M.M. Weijters, and W.M.P. van der Aalst, “The ProM framework: A New Era in Process Mining Tool Support,” In G. Ciardo and P. Darondeau, editors, *Application and Theory of Petri Nets 2005*, volume 3536 of *Lecture Notes in Computer Science*, pages 444-454. Springer-Verlag, Berlin, 2005.

[4] T. Blickle, H. Hess, “Automatic Process Discovery with ARIS Process Performance Manager (ARIS PPM)”, Expert Paper, IDS Scheer

[5] A. Rozinat and W.M.P. van der Aalst. Conformance Checking of Processes Based on Monitoring Real Behavior. Information Systems, 33(1):64-95, 2008.

[6] W. M. P. van der Aalst, A. Weijters, L. Maruster, “Workflow Mining: Discovering Process Models from Event Logs”, IEEE Transactions on Knowledge and Data Engineering 16(9),1128–1142 (2004)

[7] J. Woodfill, M. Stonebraker, “An Implementation of Hypothetical Relations”, In Proceedings of the 9th international Conference on Very Large Data Bases (October 31 - November 02, 1983). M. Schkolnick and C. Thanos, Eds. Very Large Data Bases. Morgan Kaufmann Publishers, San Francisco, CA, 157-166. (1983)

[8] M. Dumas, W. M. P. van der Aalst, and Ter Hofstede, “Process-aware information systems: Bridging people and software through process technology”, 2005: John Wiley & Sons, Inc.

[9] F. Curbera, Y. Doganata,, A. Martens, N. K. Mukhi, A. Slominski, “Business Provenance – A Technology to Increase Traceability of End-to-End Operations” R. Meersman and Z. Tari (Eds.): OTM 2008, Part I, LNCS 5331, pp. 100–119, 2008.

[10] J.E. Cook, A.L. Wolf, “Discovering Models of Software Processes from Event-Based Data”, ACM Transactions on Software Engineering and Methodology,7(3):215–249, 1998.

[11] H. R. Motahari-Nezhad, R. Saint-Paul, B. Benatallah, F. Casati, P. Andritsos, “Process Spaceship: Discovering Process views in Process Spaces, Technical Report”, UNSW-CSE-TR-0721, The School of Computer Science and Engineering, The University of New South Wales, Australia, December 2007

[12] A. K. Alves, “Using Genetic Algorithms to Mine ProcessModels: Representation, Operators and Results”, 2003.

[13] B.F. van Dongen and W.M.P. van der Aalst, “Multi-Phase Process Mining: Building Instance Graphs”, In P. Atzeni,W. Chu, H. Lu, S. Zhou, and T.W. Ling, editors, International Conference on Conceptual Modeling (ER 2004), volume 3288 of Lecture Notes in Computer Science, pages 362–376. Springer-Verlag, Berlin, 2004.

[14] R. Agrawal, D. Gunopulos, F. Leymann, “Mining Process Models from Workflow Logs”, In Sixth International Conference on Extending Database Technology, pages 469–483, 1998.

[15] S. Goedertier, D. Martens, J. Vanthienen, B. Baesens, “Robust Process Discovery with Artificial Negative Events The Journal of Machine Learning Research “, MIT Press, Volume 10, Dec 2009

[16] V. R. Basili, D. M. Weiss, “A methodology for collecting valid software engineering data”, IEEE Transactions on Software Engineering, SE-10(6):728-738, November 1984

[17] A. L. Wolf, D. S. Rosenblum. “A Study in Software Process Capture and Analysis”, 2nd International Conference on the Software Process, Berlin, Germany, February 1993.

[18] W.M.P. van der Aalst, “Process Mining and Monitoring Processes and Services: Workshop Report”, In F. Leymann, W. Reisig, S.R. Thatte, and W.M.P. van der Aalst, editors, *The Role of Business Processes in Service Oriented Architectures*, number 6291 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, July 2006.

[19] M. Castellanos, K. Alves de Medeiros, J. Mendling, B. Weber, A.J.M.M Weitjers, “Business Process Intelligence”, In: J. Cardoso andW.M.P.van der Aalst (eds.): Handbook of Research on Business Process Modeling. Idea Group Inc (2009) pp. 456-480

[20] R. Pérez-Castillo, B. Weber, I. G. R. de Guzmán, M. Piattini: Process mining through dynamic analysis for modernising legacy systems. IET Software 5(3): 304-319 (2011)

[21] R. Pérez-Castillo, B. Weber, J. Pinggera, S. Zugal, I. G. R. de Guzmán, M. Piattini: Generating event logs from non-process-aware systems enabling business process mining. Enterprise IS 5(3): 301-335 (2011)

[22] D. R. Ferreira, D. Gillblad, Discovering Process Models from Unlabelled Event Logs, Proceedings of the 7th International Conference on Business Process Management (BPM 2009), LNCS 5701, pp. 143-158, Springer, 2009

[23] A. Burattin, R. Vigo: A framework for semi-automated process instance discovery from decorative attributes. CIDM, 2011: 176-183