With the increasing automation of business processes, growing amounts of process data become available. This opens new research opportunities for business process data analysis, mining and modeling. The aim of the IFIP 2.6 - International Symposium on Data-Driven Process Discovery and Analysis is to offer a forum where researchers from different communities and the industry can share their insight in this hot new field.

Submissions aim at covering theoretical issues related to process representation, discovery and analysis or provide practical and operational experiences in process discovery and analysis. Language for papers and presentations is English. In this sixth edition, 17 papers were submitted and 13 papers were accepted for publication in the pre-symposium volume. According to the format of a symposium the discussion during the event is considered a valuable element that can help to improve the quality of the results proposed or the approach in presenting results. For this reason, authors of accepted papers will be invited to submit extended versions of their articles to a post-symposium volume of Lecture Notes in Business Information Processing, scheduled in 2017.

Our thanks go to the authors who submitted to the conference, to the board of reviewers that made an deep work, and to those who participated in the organization or in the promotion of this event.

We are very grateful to the Università degli Studi di Milano, the Graz University of Technology, and the IFIP, for supporting this event.

Paolo Ceravolo
Christian Guetl
Stefanie Rinderle-Ma
SIMPDA co-Chairs
# Table of Contents

## * Research Papers

<table>
<thead>
<tr>
<th>Title</th>
<th>Authors</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process Harmonization Phase Model in Post Merger Integration</td>
<td>Irene Schönreiter</td>
<td>pp. 3-22</td>
</tr>
<tr>
<td>Location-based Automated Process Modelling</td>
<td>Ye Zhang, Olli Martikainen, Riku Saikkonen and Eljas Soisalon-Soininen</td>
<td>pp.23-34</td>
</tr>
<tr>
<td>A Case Study on the Business Benefits of Automated Process Discovery</td>
<td>Matej Puchovsky, Claudio Di Ciccio and Jan Mendling</td>
<td>pp. 35-49</td>
</tr>
<tr>
<td>Model and Event Log Reductions to Boost the Computation of Alignments</td>
<td>Farbod Taymouri and Josep Carmona</td>
<td>pp. 50-62</td>
</tr>
<tr>
<td>DB-XES: Enabling Process Discovery in the Large</td>
<td>Alifah Syamsiyah, Boudewijn F. van Dongen and Wil M.P. van der Aalst</td>
<td>pp. 63-77</td>
</tr>
<tr>
<td>Improving Process Model Precision by Loop Unrolling</td>
<td>David Sanchez-Charles, Marc Sole, Josep Carmona and Victor Muntés-Mulero</td>
<td>pp. 89-99</td>
</tr>
<tr>
<td>Clustering Software Developer Repository Accesses with the Cophenetic Distance</td>
<td>David Sanchez-Charles, Josep Carmona, Victor Muntés-Mulero and Marc Sole</td>
<td>pp. 100-110</td>
</tr>
</tbody>
</table>
* Short Papers

Analysis of Energy-Efficient Buildings through Simulation and Formal Methods  
*Maryam Ehsanpour, Luciano Baresi, Matteo Rossi and Ernesto Damiani*  
pp. 113-119

Analysis of coordinating activities in Collaborative Working Environments  
*Muhammad Kiani and Abdul Rafay Awan*  
pp. 120-124

Categorizing Identified Deviations for Auditing  
*Marzie Hosseinpour and Mieke Jans*  
pp. 125-129

Capturing Resource Behaviour From Event Logs  
*Marijke Swennen, Niels Martin, Gert Janssenswillen, Mieke Jans, Benoît Depaire, An Caris and Koen Vanhoof*  
pp. 130-134

* Research Plans and Demonstrations

Accelerating Process Mining using Relational Databases  
*Alifah Syamsiyah, Boudewijn F. van Dongen and Wil M.P. van der Aalst*  
pp. 137-141

Students’ Modeling based on their Problem Solving Behavior  
*Rabia Maqsood*  
pp. 142-145
Conference Organization

* Conference Co-Chairs

Paolo Ceravolo
Università degli Studi di Milano, Italy

Christian Guetl
Graz University of Technology, Austria

Stefanie Rinderle-Ma
University of Vienna, Austria

* Advisory Board

Ernesto Damiani, Università degli Studi di Milano, Italy
Erich Neuhold, University of Vienna, Austria
Maurice van Keulen, University of Twente, The Netherlands
Philippe Cudré-Mauroux, University of Fribourg, Switzerland
Robert Meersman, Graz University of Technology, Austria
Wilfried Grossmann, University of Vienna, Austria

* PhD. Award Committee

Gregorio Piccoli, Zucchetti spa, Italy
Paolo Ceravolo, Università degli Studi di Milano, Italy

* Publicity Chair

Fulvio Frati, Università degli Studi di Milano, Italy
Program Committee

AKHIL KUMAR PENN STATE UNIVERSITY, USA
ANDREA BURATTIN UNIVERSITY OF INNSBRUCK, AUSTRIA
BABIGA BIRREGAH UNIVERSITY OF TECHNOLOGY OF TROYES, FRANCE
BENOIT DEPAIRE UNIVERSITY OF HASSELT, BELGIUM
CHINTAN MRIT UNIVERSITY OF TWENTE, THE NETHERLANDS
CHRISTOPHE DEBRUYNE TRINITY COLLEGE DUBLIN, IRELAND
DETLEF NAUCK BRITISH TELECOM RESEARCH, UK
EBRAHIM BAGHERI RYERSON UNIVERSITY, CANADA
EDGAR WEIPPL TU VIENNA, AUSTRIA
HARIS MOURATIDIS UNIVERSITY OF BRIGHTON
HELEN BALINSKY HEWLETT-PACKARD LABORATORIES, UK
JAN MENDLING VIENNA UNIVERSITY OF ECONOMICS AND BUSINESS, AUSTRIA
JANEZ HRASTNIK AMIS TELECOMMUNICATIONS, SLOVENIA
JOSEP CARMONA UPC - BARCELONA, SPAIN
MARCELLO LEIDA TAIGER, SPAIN
MASSIMILIANO DE LEONI EINDHOVEN TU, NETHERLANDS
MATTHIAS WEIDLICH IMPERIAL COLLEGE, UK
MAURICE VAN KEULEN UNIVERSITY OF TWENTE, THE NETHERLANDS
MEIKO JENSEN RUHR-UNI_VER_SI TY BO_CHUM, GERMANY
MOHAMED MOSBAH UNIVERSITY OF BORDEAUX
PNINA SOFFER UNIVERSITY OF HAIFA, ISRAEL
ROBERT SINGER FH JOANNEUM, AUSTRIA
ROLAND RIEKE FRAUNHOFER SIT, GERMANY
THOMAS VOGELGESANG UNIVERSITY OF OLDENBURG, GERMANY
WIL VAN DER AALST TECHNISCHE UNIVERSITEIT EINDHOVEN, THE NETHERLANDS
MARK STREMBECK WU VIENNA, AUSTRIA
MARIA LEITNER AUSTRIAN INSTITUTE OF TECHNOLOGY, AUSTRIA
Research Papers
Process Harmonization Phase Model in Post Merger Integration

Irene Schönreiter
Technical University Dresden, Germany
Irene@schoenreiter.de

Abstract. Post merger integration (PMI) has received much attention in recent years due to an increasing number of merger and acquisitions (M&As). Process harmonization plays an important role during the PMI. The objective of this article is to define the milestones of process harmonization and develop a phase model. Nine approaches are illuminated and concluded to a process harmonization phase model. BPM implementation or optimization literature has been scanned for suitable approaches. Further expert interviews have been conducted and evaluated with qualitative content analysis. A combination of the different approaches is regarded as the optimum. This article provides two central results: First, process harmonization phases are divided into management system level and process level. Second, process harmonization phases exist of analysis phase, conception phase, realization phase and verification phase. A general overview of suitable methods for each phase is provided.

Keywords: process harmonization · post merger integration · phase model

1 Introduction

“The integration process is the key to making acquisitions work“ [1]. Due to the fact that Business Process Management (BPM) plays a vital role in organizational changes [2, 3], a key element in post merger integration (PMI) is the process harmonization (PMI). The relevance of BPM for practioners is stagnantly growing [4]. Especially when symbiosis or absorption is the company’s chosen integration approach, two worlds of business processes and management systems have to be harmonized efficiently.
This article intends to suggest an approach for the phases of process harmonization in the PMI phase. Process harmonization typically comes along with process optimization and process redesign. Process redesign is frequently a way for improving process performance dramatically [5]. An essential prerequisite for a systematic design and optimization of results, processes and resources are standardized processes and a unified process framework as a basis for mastered processes [2]. So why not to use the chance of improving processes when they are redesigned by harmonisation anyway? The way from separate companies to efficiently harmonized and optimised processes includes some milestones and is divided in several phases.

This article aims to find an answer for the following research question: Which phases can be used for process harmonization in post merger integration phase?

The paper is structured as follows. After this introduction, the theoretical background is explained followed by the identification of relevant literature which is added by expert interviews. Chapter 4 presents the results that are discussed in chapter 5. A summary, limitations and ideas for future research close this article.

2 Theoretical Background

Fernandez & Bhat define process harmonization as “process of designing and implementing business process standardization across different regions or units so as to facilitate achievement of the targeted business benefits arising out of standardization whilst ensuring a harmonious acceptance of the new processes by different stakeholders” [6]. Process harmonization aligns the differences of standardized processes and defines the degree of their variation [7, 8] “by capturing
their commonality and variability in a consolidating and efficient manner, without attempting to make different processes identical" [8].

Aside from post merger IT integration there is no identified methodology to harmonize processes in the post merger situation [9]. Post-merger integration refers to the integration of a company after the signing of a M&A in which the integration planning and implementation takes place in order to realize the desired appreciation successful [10].

According to the need for organizational autonomy and need for strategic interdependence, different types of integration approaches exist: absorption, symbiosis, preservation and holding [1, 11, 12].

- Absorption: the acquired company is absorbed by acquirer and assimilated into its culture, the management usually comes from the acquirer [13],
- Symbiosis: evolution from existing [14], learn from each other and share qualities [12], most complex managerial challenge [1]
- Preservation: acquired company retains independence, modest degree of integration by acquired company [13], no novelty [14]
- Holding: integration is not intended [1]; the acquired company usually exists as a separate legal entity [15].

For process harmonization are especially absorption and symbiosis of relevance.

3 Phases for Process Harmonization

3.1 Identification of Relevant Literature

To discover the optimal phases for process harmonization there are various starting points.
Changes coming along with process harmonization can generally be proceeded evolutionary or revolutionary. The evolutionary change corresponds to the process of continuous improvement and is a gradual, targeted and continuous approach, in which the corporate structure and strategy is maintained [16]. Total Quality Management (TQM) and Six Sigma can be associated with the evolutionary approach [17]. The evolutionary approach equates to the engineering of processes and consists of as-is analysis, as-is modeling, target modeling and process optimization [18]. The advantage persists of a low risk of not revisable wrong decisions in the rather complex issue [15]. If there are already processes with a high level maturity in a company, it is disadvantageous, destroying them by the radical approach, rather than develop [19].

The radical approach means Business Process Reengineering (BPR). BPR defined in short means “start from scratch” [20] and thus corresponds in its pure form the "green field" approach [19]. BPR or shortly Business Reengineering (BR) is a method for process optimization done solely top-down and requires the absolute backing of top management [21]. This involves the complete redesign of processes and procedures in the company with the target to increase in performance in terms of cost, quality, service and speed [22]. BPR proceeding is following: 1. Why is something done?, 2. What needs to be done for?, 3. Clarification of strategic specifications and target framework, 4. Elimination of old processes and structures, 5. Distinguish process and new design of today’s sight, 6. Consideration of BPR principals, 7. Radical new designed process (to-be concept and implementation plan), 8. Vernier adjustment and stabilisation [22].

Another starting point for the harmonization of integrated management systems introduces Karapetrovič that might be adaptable in the post merger integration for two different existing management systems. Ba-
based on the system-oriented approach, the harmonization is achieved by the following steps: 1. Integration of the documentation in a common manual and otherwise separated process, 2. unification of the main processes, objectives and resources, 3. "All-in-one-system" as a universal system for all sub-systems - complemented by the development of a common audit system [23]. A study of Sampaio et al. examined the question of whether integration or supplementation of the additional management systems to an Integrated Management System. The investigation showed following levels of integration: Integration of documentation, management tool integration, definition of common quality policy and objectives, common organizational structure [24]. Ntungo developed the "Pursuit of Excellence Quality Management System (PEQMS)" with following steps and ISO 9001 as frame requirement: 1. Development of the quality policy 2. Understanding quality of assessment and taxation services 3. Developing quality objectives 4. Identifying and understanding business processes 5. Developing and implementing quality standards 6. Using quality data for continuous improvement 7. Sustaining the quality management effort 8. Option to register with the ISO 9001 standard [25].

Han et al. propose a two-stages business process analysis model for increasing the efficiency and effectiveness of business processes: the macro process analysis specifies at the business performance with the highest influence and defines a to-be standard. The second stage – micro analysis – uses process simulation and consists of a review of the as-is-process and designs a to-be process [26].

Another approach to harmonize management systems might be the adaption of a new implementation project. Schmelzer & Sesselmann define following phases for the introduction of BPM: Positioning, Identification, implementation, optimisation. [27].
Seven approaches with various phases have been identified in literature. In the next section experts are interviewed about process harmonization phases in practice.

### 3.2 Qualitative Analysis

A qualitative analysis is used to identify the procedure of process harmonization in practice. Therefore twelve experts composed of CEOs/COOs (33%) and quality managers (67%) of different service sector companies have been interviewed and asked about their experience in process harmonization. A qualitative semi-structured interview has been conducted with experienced experts in process harmonization after M&A. The evaluation was performed by qualitative content analysis according to Mayring [28, 29]. Interviewing experts is aligned by specific problems and demand of a questionnaire and offers the interviewee an extended space for the answer [30], so the orientation is more subjective and interpretative [31].

The experts have been requested to describe the procedure of process harmonization in their company after an M&A. They specified their individual situation, good and bad experiences with the described procedure and – most important – their satisfaction.

Appendix 1 presents the procedure of choosing the relevant cases for harmonization phases in practice. In this article two cases out of the twelve are regarded: both cases have a satisfactory result and the interviewed expert would apply the same procedure again; some of the experts are quite dissatisfied, so for identifying process harmonization phases best practices are needed; the used integration approach was symbiosis or absorption; both cases are finished and long-term experience is available.

Expert 2 advises following procedure to include all merged companies in a common certification: 1. Definition of cramp processes for certification (coverage of minimum norm requirements), 2. Integration of common quality culture (regulations, dynamic, reporting, …), 3. Definition of responsibilities and interfaces (work groups existing of quality management and single departments), 4. as-is record of processes with swimlane diagram, 5. Step-by-step unification of processes with room for individual processes, 6. Unification of IT systems, e.g. ERP system, common documentation database, 7. Implementation, 8. Verification by internal audits.

3.3 Analysis and Interpretation

Table 1 summarises the various proceedings presented in the literature review in chapter 3.1 and qualitative analysis in chapter 3.2. Number 1-7 present the approaches identified in the literature review and number 8+9 show two cases of the expert interviews, so each row describes one approach. The title gives a short description of the approach, detailed information are described in Appendix 2.

<table>
<thead>
<tr>
<th>Nr.</th>
<th>Author</th>
<th>Title</th>
<th>Nr.</th>
<th>Author</th>
<th>Title</th>
<th>Nr.</th>
<th>Author</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[27]</td>
<td>Introduction of BPM</td>
<td>4</td>
<td>[25]</td>
<td>PEQMS</td>
<td>7</td>
<td>[24]</td>
<td>Integration levels of integrated management</td>
</tr>
</tbody>
</table>
These nine harmonization approaches express a quite different picture at first sight. They differ in number of steps as well as in a clear starting and end point. The proceedings of Schmelzer & Sesselmann and the recommendation of expert 2 demonstrate the steps of a whole process lifecycle defined by Scheer & Hoffmann. The BPM lifecycle is divided into following phases connected in a cycle: Process Strategy, Process Design, Process Implementation, Process Controlling [32]. Karapetrovic, Ntungo, Gaitanides, Sampaio et al., Jochem & Geers and expert 1 concentrate on the development of a management system, whereas Han et al. is focused on the analysis phase.

4 Results

4.1 Analysis Phase

Concepts 6, 8, 9 suggest directly an as-is analysis, partly combined with an as-is modelling of the processes. Further concepts let conclude to an analysis phase as well. N° 1 (identification), 2 (requesting why is something done and what needs to be done for) and understanding the processes of concept 4 are also part of an analysis phase. Hence in conclusion for understanding the processes of both merged companies, an “analysis phase” is identified as first phase. The approach of micro and macro analysis of Han et al. (2009) will be taken over and adapted to process harmonization. As there is needed the consideration of the whole management system towards overall
regulation and control and the compliance with company objectives and strategy, a macro perspective is necessary for process harmonization. On the other hand a micro perspective on the single process level is required. Each single process has to be regarded for the unification of both, the available same process of the merged company part and on the compliance with the macro level, means management system. Hence each phase of process harmonization is regarded on management system level and on process level. Regarding the steps of the nine approaches, they jump between management system and process level, too.

It is imperative to observe the maturity of business processes, before any re-designing or optimisations to a higher level are started [5]. So on management system level a maturity assessment is constituted for an overall analysis, whereas on process level a systematic process analysis is advised.

4.2 Conception Phase

Next the “conception phase” is identified. All concepts except N° 5 have conceptual steps. Process optimisation, changed process design, to-be modelling, definition of processes, the definition of responsibilities and process interfaces and the unification of IT systems are part of the concept phase on process level.

The development of new quality standards, policy and objectives, strategic specifications, the development of a new structure, the integration of a new quality culture are all contents of conceptual work on an overall management system basis – in compliance with the organisations’ strategy. As already identified in the analysis phase, the conception phase is also divided into a management system level and pro-
cess level. As strategic alignment is often identified as critical success factor in business process management [4, 33] and even defined as one of six core elements of BPM by vom Brocke & Rosemann [34], the strategic alignment shall be a first important key element for the management system that needs to be compliant with the organizations strategy.

On process level Wirtz presents depending on the congruence and efficiency of processes following possibilities for harmonization, which differ in the level of integration intensity: standardization of processes, adaption and optimization of processes, redesign of processes [15]. If processes differ significantly, but possess high efficiency, the standardization/unification of the processes is advisable. Similar processes can be applied of the more efficient corporate part. If process standards are low, the processes should be redesigned [35]. If processes are largely congruent and efficiency high, the process can be taken over and tuned if necessary. A subtle balance between integration and differentiation must be found in the harmonization [16]. A too rigid integration can restrict the flexibility of employees and block on specific requirements. To avoid useless “overstandardization” Schäfermeyer & Rosenkranz recommend manager to differ carefully between routine processes with a low complexity and nonroutine processes with a high complexity to identify standardable processes and save time investments for non-standardable processes [36]. Romero et al. come to a similar result: companies that have less complex processes have more harmonized processes and also more standardized processes [8, 37].

To support the management in priorization and selection of suitable business processes for standardization, Zellner & Laumann developed a decision tool influenced by various process characteristics [38].
4.3 Realization Phase

After the development of a harmonization concept, things have to get run and implemented. Although some of the considered approaches stop with the development of the concept, the implementation is a vital step in process harmonization. So the “implementation phase” is identified as third phase. In respect to supposed shared responsibilities on management system level and process owners this phase is again divided into management system level and process level.

The chronological introduction of the adapted processes and management system has to be decided. A step-by-step implementation with a gradually iterative introduction in one department after another takes longer, but reduces the risk and complexity of the project. In contrary implementing the new concept into force on "Day X" (Big-bang) means the simultaneous introduction in the entire company with the advantage of high speed integration, but with the risk of a high failure rate. A pilot-operated launch in one department is time-intensive and causes additional administrative expenses through parallel worlds, but failures are not repeated [39].

4.4 Verification Phase

The last phase is the “verification phase”. Approach N° 4 focuses on quality management sustainability and continuous improvement, N° 9 verifies the implemented processes with internal audits, so the verification is regarded as an effective closure of process harmonization. The success of the harmonized processes and of the management system can be judged in this phase.

Expert 2 suggests the verification of process harmonization through internal audits. A combination of system audit and process audits
measure the success of both levels, management system and the single processes. With the combination of process controlling in internally defined process KPIs and performance measurement of the process output the verification can be measured effectively.

Process management maturity assessments shall be repeated periodically to get an overview of the development of continuous improvement. With the maturity of a process and of the management system compared with the origin maturity result during the analysis phase, the verification is achievable as well. Using the same maturity model allows a direct comparison. A better maturity result for the same process or the management system as in the analysis phase evidences a successful conception and implementation.

Fig. 1. shows the process harmonization phases in an overview, divided in management system (MS) level and process level.

![Fig. 2. Process Harmonization Phase Model](image)

5 Discussion

Given the situation of a symbiosis or absorption, the organisation is interested in assessing its current business processes and management system situation. For the analysis phase maturity models are required for both, the double existing management system and each double existing single process of the merged organizations. Maturity of processes and management system might be quite different.
None of the experts mentioned the maturity assessment during the analysis phase explicitly. Most of them presumed the comparability of the double existing processes and management systems as self-evident. One expert described the procedure as the use of “horse-sense”. Hence the experts used no defined and well-known methodology for comparing and analysing the business processes. As a defined methodology conveys neutrality and professionalism to an often tense situation with divided interests, the application of a maturity model is strongly recommended. One merged company part might hold a very mature and stable management system, but single processes are superior in the other part. To differentiate between management system level and considering the maturity of each process as well, leads to an enhanced result. The premise of course is the sufficient consideration in the conception phase. Results are more accurate if different levels for the assessment are involved. Middle and low management tend to be more critical than top management and operational staff [40].

Hammer (2007) encourages assessments with different staff level as well to illustrate possibly existing homogeneity in understanding of processes and assessment criteria [5]. A 360-degree-feedback with involvement of customers and suppliers is conceivable [40].

In context with the implementation phase the integration speed is a controversial issue. On the one hand, a high integration speed ensures competitiveness due to a relatively short state of excitement; on the other hand, blockers within the organization might take the time to act [42]. 25% of the experts state in their best practices they prefer a longer period of time to get to know each other, and thus a slower integration speed. This affects also the implementation strategy. As shown in chapter 5.3 the big-bang implementation is the quickest variant compared with step-by-step or pilot introduction. The variant to be pre-
ferred has been defined individually in the merged companies and aligned to the general PMI strategy.

Process harmonization is a complex project. The more significance has the continuous strategic view on the whole project on a management system basis. A harmonized process past on the defined strategy and non-conform with the overall management system is not expected to perform as a whole. So the combined consideration of process level and management system level as well as strategic alignment is essential. This does not mean that individual variations in business processes are not allowed. In contrary, the experts agree upon the necessity of reasonable process variations, e.g. for the fulfillment of a special customer requirement necessary at one department only. Process harmonization does not try to make standards uniform, it looks how standards fit together [7]. In the conception phase the trade-off between process standardization and process variants has to considered carefully, although with the number of variants the process harmonization is measurable [41].

A third of the experts state a good satisfaction with the process harmonization results, although most of them would apply a different procedure next time. So the long-term results can consequence in a positive satisfaction with the process harmonization procedure after a series of corrective actions. The main part of the interviewees shared lessons learned initiated by difficulties during the harmonization project. It might be assumed that companies need more than one M&A before they proceed post merger integration successfully. The milestones identified in the process harmonization phase model at hand are a key prerequisite for executing a successful post merger process harmonization.
6 Summary, Conclusion and Limitations

The objective of this article is the definition of a phase model for process harmonization in the post merger integration phase. Various approaches of the literature have been scanned and complemented with a qualitative research to obtain the milestones of a process harmonization. As a central result can be concluded that there is no special methodology with defined phases for process harmonization in literature available, so a combination of the different approaches is regarded as the optimum for process harmonization. This article provides two central results: First, process harmonization phases are divided into management system level and process level. Second, process harmonization phases exist of analysis phase, conception phase, realization phase and verification phase. Further this article presents a general overview of possible methods for each single phase.

The author would like to indicate to a limitation concerning the expert interviews. Other experts might have different best practices. However both are typical examples of the executed qualitative analysis, so the cases are regarded as sufficient. Next the mentioned methods are not evaluated empirically. The purpose of this article was the definition of the phases and not the definition of methods within the phases. So as a next research step proper methods for the defined phases should be evaluated empirically.

7 References

12. Marchand, M.: When the south takes over the north: Dynamics Of up-market integrations by emerging multinationals. M@n@gement. 18, 31–53 (2015).
## Appendix 1

### Table: Results Expert Interviews

<table>
<thead>
<tr>
<th>Question in interview guideline</th>
<th>Variable</th>
<th>Variable characteristic</th>
<th>Result</th>
<th>Relevance for process phases</th>
</tr>
</thead>
</table>
| How did you approach the process harmonization? | Integration approach | Symbiosis, absorption, preservation, stand-alone | Absorption: 27%  
Symbiosis: 20%  
Stand-alone: 7%  
Preservation: 46% | Relevance: included in case study in chapter 3.3 with symbiosis or absorption approach only |
| Please explain how the process harmonization was expired. | Procedure | Description of interviewee (open answer) | individual description for each case | Identification of main phases on general level |
| How was the harmonization carried out on the process level, in the individual process? | Procedure process | Description of interviewee (open answer) | individual description for each case | Identification of main phases on single process level |
| How satisfied are you with the result of the process harmonization? | Satisfaction | Very satisfied, satisfied, undecided, unsatisfied, very unsatisfied, not stated | Very satisfied: 3  
Satisfied: 1  
Undecided: 4  
Unsatisfied: 1  
Very unsatisfied: 1  
Not stated: 2 | Relevance: included in case study in chapter 3.3 only with good or very good satisfaction |
| What would you do differently if you start the project again? | Lessons Learned | Exactly the same, different procedure, not stated | Not stated: 5  
Exactly the same: 2  
Different procedure: 5 | Relevance: included in case study in chapter 3.3 only if expert would proceed in the same way again |

The first column shows the questions asked to the experts. All questions are open questions, so the answers of the experts are quite diffe-
rent in content and length. Each question has been coded according to column “variable” with the possible content described in column “variable characteristic”. Column “result” shows the result after encryption. The two questions for describing the process harmonization was evaluated individually, the answers could not be coded unified. All other answers have been summarized according to the variable characteristic. The last column “relevance for process phases” describes the importance of each row for the following selection of the case studies. Only the combination of absorption or symbiosis approach with a satisfactory result and the statement of an expert he/she would proceed in exactly the same way again, qualifies the case to be regarded in chapter 3.3.
Appendix 2

Table: Summary of Approaches

<table>
<thead>
<tr>
<th>No</th>
<th>Author</th>
<th>Title</th>
<th>Steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>Interview expert 2</td>
<td>Common certification</td>
<td>1. Definition of common cramp processes</td>
</tr>
</tbody>
</table>

The nine approaches contain certain phases, described as steps in the fourth column.
Abstract. Services are today over 70% of the Gross National Product in most developed countries. Hence, the productivity improvement of services is an important area. How to collect data from services has been a problem and service data is largely missing in national statistics. This work presents an approach to collect service process data based on wireless indoor positioning using inexpensive wireless sensors and smartphones. This work also presents how the collected data can be used to extract automatically the process model. These models can further be used to analyse the improvements of the service processes. The presented approach comprises a light-weight process data acquisition system, which collects a minimised but precise data sets for automated process modelling. This automated modelling can be used to greatly improve the traditional process modelling in various service industries, for example, in the healthcare field. The presented approach has been tested and used in Tampere City dental care clinics.

Keywords: automated process modelling, process mining, location-based

1 Introduction

Service intelligence is becoming a worldwide trend. Efficiently and effectively running service operations are the key for gaining a competitive edge in almost every industry. The implementation of service intelligence relies heavily on a deep understanding of the service process, however, how to collect data from services has been a problem. This work presents an approach to collect service process data based on wireless indoor positioning using inexpensive wireless sensors and smartphones.

A service process is a set of activities in a chronological order and outputs a service as the final product. In this work we model processes graphically using boxes and arrows. One box represents an activity with service time and arrows indicate the transitions between activities. Based on the process data that we acquire in this work, we measure the average service time of each activity, and also transition probabilities between activities. This work focuses on modelling the generic service processes. Usually this type of services is location-aware, which means activities happen in specific locations. Therefore, we figured out that location information can be used to infer activities of generic service processes. Figure 1 illustrates our approach, which includes 4 phases:
a). *Design phase*, it requires manually determining targeted activities and corresponding locations. Then plan the setting of wireless sensors and attach them to locations specified by activities.

b). *Calibration phase*, it trains a set of measurement sensors on the mobile device side. Besides, it collects training data and transfers them to the server side for computing activity patterns and other parameters.

c). *Process measurement phase* acquires process data and synchronises them to the server continuously for activity recognition.

d). *Process modelling phase*, it models the whole process on the server side, based on the information of recognised activities.

![Fig. 1: The approach of real-time automated process modelling](image)

We had extensive involvement in modelling service processes and improving service qualities in the healthcare sector. We have cooperated with Helsinki University Hospital, Meilahti Hospital and Tampere Dental Clinic. We used to model processes based on interview data [6], and then based on the captured process model, we analysed process performance optimisation with a tool called 3VPM [8]. However, social consulting agencies featured that automated wireless measurement as a more cost effective approach. This was the reason for starting our research. The idea of modelling location-aware processes was initiated in our previous work [21]. Nevertheless, the prototype in Zhang et al. [21] wasn’t feasible for automated generating process model, activity analysis and process modelling was done manually. In addition to simple location data we have previously researched pattern recognition of signal sequences from wireless sensors attached to places and people to identify activities. These pattern recognition
techniques have been patented [9]. This work is an extension of our previous research and aims to improve the quality of the obtained location data and implement automated process modelling. This work contributes to the following aspects:

a). Proposing an approach for automated process modelling,

b). Implementing a light-weight process data acquisition system by utilising Bluetooth indoor positioning technique and Internet of Things (IoT).

This paper is organised as follows. Section 2 discusses related works. Section 3 illustrates the process data acquisition system and the analytical approach of process model extraction. In Section 4, we evaluate the system in a laboratory case study. Section 5 concludes the paper with the limitations of the current system and directions for future work.

2 Related works

The key factor of implementing service intelligence is successfully modelled service processes. Halonen et al. [6] documented process models extracted from interview data, and then used them in process performance optimisation. Based on a comparative analysis of four Australian public hospitals’ healthcare processes, Partington et al. [15] demonstrated that through analysing the processes, it provided detailed insights into clinical (quality of patient health) and fiscal (hospital budget) pressures in health care practice. Another research [18] used declarative models for describing healthcare processes and reported that process mining can be used to mediate between event data reflecting the clinical reality and clinical guidelines describing best-practices in medicine.

Process mining has been widely explored in the healthcare sector, Halonen et al. [6] structured the processes in the acute neurology ward of Helsinki University Hospital by collecting data from process personnel interviews. Other research focused on analysing event logs of existing administrative systems or medical devices. Usually their targets are to solve a particular problem [11]. For example, Rebuge et al. [17] analysed the hospital emergency service, Mans et al. [10] studied the gynecological oncology process, Blum et al. [2] mined laparoscopic surgery workflow. However, approaches that are capable of picturing more generic process are still missing. We found out that generic processes usually have no trails in existing event logs. Accordingly, we abstracted generic activities to a location-based level and integrated Bluetooth indoor positioning and Internet of things techniques in the procedure of process modelling.

We are currently in the Big Data era, it opens new prospects for every industry and it is indeed promising to enable service intelligence [12] [19]. Nonetheless, the integration of high volume data from various sources complicates the operation of process mining. Moreover, we have learned from studies [3], the data quality of real-life logs is far from ideal, they are usually noisy, incomplete and imprecise. As a result, we were facing challenges such as how to guarantee the quality of data used for process modelling. Therefore, we intended to simplify the
procedure of automated process modelling by developing a light-weight process data acquisition system that collects minimised, but precise data sets.

In order to collect process activity related location data, we decided to use Bluetooth as an indoor positioning technique. Bluetooth has the advantages of low cost, highly ubiquitous, low power consumption, ad-hoc connection and shorter-range with room-wise accuracy [7] [4], which suits our purpose very well. Bluetooth in indoor positioning is a mature research field and has been widely studied [1] [14]. We utilised location information to infer abstract activities performed in corresponding locations besides using Bluetooth purely as indoor positioning technique. From this perspective, our research is closer to Faragher and Harle [5], which used fingerprint techniques such as pattern matching approaches to recognise activities. Faragher and Harle [5] declared that due to the instability of radio signal propagation, pattern matching approaches are much more effective than approaches that use radio proximity.

The developments in inexpensive and unobtrusive sensors, machine learning and data mining techniques have enabled automated process modelling. Wan et al. [20] used an online sensor data segmentation methodology for near real-time activity recognition. Okeyo et al. [13] presented an approach to real-time sensor data segmentation for continuous activity recognition. Pham et al. [16] implemented a real-time activity recognition system to detect low-level food preparation activities. Likewise, they also streamed sensor data continuously for real-time analysis. Nonetheless, most of them focus on detecting motion activities and there are not yet enough application of IoT in recognising process activities. Differentiate from discrete motion activities, our targeted activities are process related activities, which are usually ordered and have transition probabilities between them.

3 Automated process modelling system

With the objectives to guarantee the quality of data used for process modelling, meanwhile to simplify the procedure of the generic healthcare process modelling, we propose an automated process modelling system. The system consists of four principal modules: a). Process data acquisition, b). Calibration of the process measurement, c). Process measurement, and d). Process model extraction.

3.1 Process data acquisition

The process data acquisition module intends to acquire minimised but precise activity data sets. Bluetooth data was collected to infer indoors location information, and furthermore used to represent location-based activities. Accordingly, we collect chronological sequenced tuples that measure Radio Signal Strength Indications (RSSI) of Bluetooth sensors. There are a variety of input data for process measurement. One feature of the system is its capability of measuring multiple processes. We defined a process as a measurement site that uses a specific set of Bluetooth sensors. Hence, we categorised input data into two types.
a). General input data: a full list of Bluetooth devices \( D = \{ Sensor_1, Sensor_2, Sensor_3, \ldots, Sensor_N \} + \{ User_1, User_2, User_3, \ldots, User_N \} \), which includes both Bluetooth beacons and user devices. This type of information is kept on the server side, the minimised information of a Bluetooth device needed is the Bluetooth MAC address.

b). Measurement site specific input data:
- A subset of Bluetooth devices \( Sensor_{sub} = \{ Sensor_4, Sensor_7, \ldots, Sensor_X \} \). One or multiple devices are set in a location to represent an activity, therefore, different measurement sites have different Bluetooth devices subset. Each subset is independent, but the included Bluetooth devices can be either exclusive or overlapping.
- A subset of users involved in the specific measurement, \( User_{sub} = \{ User_1, User_3, User_6, \ldots, User_Y \} \), similarly to Bluetooth devices subset, each user subset is meant for a specific process and same user can take part in multiple processes.
- A list of activities in the specific process, \( Activity_{list} = \{ Activity_1, Activity_2, \ldots, Activity_Z \} \). Each activity item is a vector that provides minimised information such as \( Activity_{ID, activityName, locationID, [Sensor_1ID, Sensor_4ID \ldots]} \), which contains activity ID, activity name, location ID and a list of sensor IDs.

In addition, we defined following attributes to be measured, a record tuple at time \( T_t \) is:

\[ Tuple_{T_t} = (T_t, User_u, [RSSI_{Sensor_1T_t}, RSSI_{Sensor_2T_t}, \ldots, RSSI_{Sensor_ST_t}]) \]

where \( T_t \) is tuple’s timestamp; \( User_u \) is the user involved in the process, it’s the Bluetooth MAC address of the user mobile device. At each time point \( T_t \), there is a \( S \)-sized RSSI vector. \( S \) is the number of Bluetooth transceivers in a particular process measurement. If Bluetooth sensor is out of range, RSSI = 0, otherwise RSSI equals the real-time measured value.

3.2 Calibration of the process measurement

In this research, we built Bluetooth transceivers with JY-MCU Bluetooth wireless serial port modules. Generally, the radio propagation is extremely complex and unstable. We tried to compare the performance of our transceivers by measuring each from the same distance. However, the radio signal strength obtained varied dramatically. As a result, we had to include an essential step in the lightweight system: calibration. It collects a training data set and generates parameters of sensor performances. Calibration only requires an administrator role to walk through all the locations with all of the user devices and let the devices measure a few data points of RSSI information at each location. This subject-independent approach to keep general users away from the burden of

training phase and provides them with a ready-to-use application. Furthermore, it will facilitate the application of this system. Responsibilities of calibration are as follows:

a). From the full list of all predefined Bluetooth devices, it trains a subset of Bluetooth devices for measuring a specific process.

b). Synchronises device's local time with the remote server time, in order to maintain the consistency of timestamps of records collected from different devices. This is an essential step for multi-users collaborative activity recognition.

c). Collects a training data set: tuples $T_{i}^{p}$, in which $i$ is a discrete time point with sampling rate interval(s), $i = 0 + n \times rate$. Thereafter, this time-based training data set is combined with user-supplied information of actual activity to form activity patterns: that is, for each activity, we collect a set of possible RSSI patterns seen in that location.

3.3 Process measurement

After selecting a particular process to measure, information about corresponding set of Bluetooth sensors will be synchronised from the remote server. Subsequently, the background service applies asynchronous Broadcast Receiver schema to periodically detect RSSI vectors. It is user interfaces independent and non-obtrusive for general users. The asynchronous broadcast receiver schema is basically a broadcast receiver keeps listening to two actions: Action One, a remote Bluetooth sensor found; Action Two, one Bluetooth inquiry finished. Action One is triggered when the mobile device enters the radio proximity of a fixed Bluetooth transceiver, in the meantime, the system collects real-time Bluetooth RSSIs. Action Two is triggered when one Bluetooth inquiry duration ends (about 12 seconds). Thereafter, a new Bluetooth discovery will start.

The sampling rate is 12s, same as the duration of one Bluetooth inquiry. Upon this architecture, the integration of IoT enables automated process modelling: the system collects tuples continuously, meanwhile, the mobile device periodically synchronises tuples to the remote server through Wi-Fi. The synchronising rate is adjustable based on the measurement needs. The system applies Google Volley networking framework to stream data between the server and mobile devices. On the server side, it applies activity pattern matching. In addition to this, the system uses sensor performance parameters to determine ambiguous activities. Ultimately, window size is used over incoming tuples to eliminate noisy activity detections.

3.4 Analytical approach for process model extraction

This work applied the following analytical approach to identify process activities and extract process model. To simplify the following discussion, we assume that

\[ \text{https://developer.android.com/training/volley/index.html} \]
there is only one person whose activities are being measured. The system is capable to measure multiple independent processes separately at the same time. The analytical approach is shown in Figure 2 and Figure 3.

![Diagram](image)

**Fig. 2: The analytical approach for process model extraction**

- a). During the training/calibration phase, two operations are performed: mobile side executes training and generates training data, which includes tuple timestamps and RSSIs of the full list of Bluetooth devices. The other operation is manually record the training process, contains information such as activity ID, activity begin and end time. Then, together with the activity information/sensor settings defined in 3.1 b), server side’s analytical process will compute activity patterns and other activity related parameters as output. In activity pattern, 0 means none of the sensors that represent this activity is in range. 1 means we can detect the radio signal of one of the activity’s sensors. 2 indicates both of the activity’s sensor are in range. In practice, there is signal overlapping issues, for example, activity pattern "A1: 1,0,1,0" means during activity 1, the device also received RSSI from sensor that represents activity 3.

- b). Process measurement collects real-time measurement data. Similar to training data, it contains tuple timestamps, but instead of RSSIs vectors of the full list of Bluetooth devices, it only records RSSIs vectors of the subset Bluetooth devices.

- c). Server’s analytical process applies activity pattern matching on each measurement tuple transferred from mobile devices. Since different activities may...
have the same patterns, a tuple may match more than one activity classes. To address this, we first detected activities that have clear activity patterns. Afterwards, we took a step further to analyse ambiguous tuples. Parameters computed in calibration, such as sensor’s mean RSSI, average performance sensors in each activity, tolerant range of signal strength for each activity are integrated to determine ambiguous activities.

d). Server applies a window size over a few \( Q \) successive activities, in order to eliminate noisy activity detections (for example, switching to a new activity momentarily, and back to the original activity in the next data point), as shown in Formula 2 in Figure 3. Generally, we assume this type of noise contains less than two tuples.

e). Next, we identified the beginnings and endings of activities. It output changes of activities, 0 means no change, -1 indicates no activities or constant change between activities, as shown in Formula 1 in Figure 3. At last, we computed average service time and transition probabilities to model the process.

![Example measurement data](image)

**Fig. 3:** Example of measurement data & analysis visualization. (Colum "Entry" is the final output of this analysis)

### 4 Case studies

We implemented the data acquisition system for Android smartphones. In addition, we evaluated the system and the analytical approach for process model extraction in a laboratory case study. We placed 17 Bluetooth transceivers in 8 locations in the computer science building in Aalto University to represent 9 activities. Figure 4 shows the setting of sensors in the process measurement. As mentioned, the performances of sensors vary and are neither stable nor consistent. In order to find an optimal setting of sensors, we conducted several
experiments and found out that, the using of two sensors to represent one activity helps improve the process measurement results. For comparison reasons, we wrote down the actual process on paper manually in addition to the automated process measurement with mobile devices. The process measurement results are presented in Figure 5a and Figure 5b.

Figure 5a shows the result of using proximity detection approach. It indicates two problems: one, when the locations of two activities are relatively close to each other, this approach will lead to noisy fluctuations; second, when there is only a very short interval between two activities, it won’t be accurate enough to determine the interval. Figure 5b demonstrates the application of the analytical approach illustrated in section 3.3. By comparing with the actual process, the result shows that the analytical approach for process model extraction detected the correct activity in 93% of the data points. It other words, the system fulfils the demand of collecting precise process data for accurate process modelling.

The process model captured from the case study is shown in Figure 6. The average service times and transition probabilities are calculated from the analysed data (i.e. the begin and end times of each occurrence of an activity) as follows. For an activity \( i \in \{1, \ldots, n\} \) that occurred \( m_i \) times in the data, the average service time is \( S_i = 1/m_i \sum_{j=1}^{m_i} d_{i,j} \), where \( d_{i,j} \) is the duration of the \( j \)th occurrence of activity \( i \). We then compute a matrix of how many (directed) transitions occurred between the activities: \( T_{i,j} = \) number of transitions from activity \( i \) to activity \( j \). From this we can calculate transition probabilities by scaling with the total number of outgoing transitions from an activity. That is, the transition probability \( P_{i,j} \) from \( i \) to \( j \) is \( P_{i,j} = \frac{T_{i,j}}{\sum_{k=1}^{n} T_{i,k}} \).

Fig. 4: Case study in Aalto University: sensors setting for process measurement
5 Conclusions

Process modelling is a critical factor in the improvement of service productivity and in the implementation of service intelligence. However, how to collect data from services has been a problem. This work focused on automated modelling of generic service processes that are location aware. In other words, activities in the process usually happen in a particular location and location information can be used to infer activities. Accordingly, we presented an approach to collect service process data based on wireless indoor positioning using inexpensive wireless sensors and smartphones. The objective of this work was to simplify the procedure of automated process modelling. For this reason, we designed a process data acquisition system to acquire minimised, but precise data set, instead of taking overwhelming redundant data. In our approach, Internet of things is integrated to implement real-time automated process modelling. We illustrated the analytical approach for process model extraction in this system and we examined the performance of the process data acquisition system and the analytical approach.
in a case study. The results of the case study demonstrate that the system fulfils the demand of collecting precise process data for accurate process modelling. In addition, the presented approach has been tested and used in Tampere City dental care clinics. Their measurement results confirm the feasibility of this approach in process modelling and the feasibility of using the extracted models in process performance optimisation.

Application status of the current system is limited to relatively ideal settings: one location represents only one activity. Besides, the system requires that two locations have a certain distance (minimum 2 meters). As illustrated in our analytical approach, we eliminate out noisy activities that have less than two tuples. Hence, the shortest activity that can be detected has at least two tuples (about 24 seconds). The current system is applicable for analysing the process of a singular user rather than analysing the collaborative process of a team. Therefore, our objective of future research is to implement automated process modelling for team collaboration process. Moreover, improve the accuracy of process activity recognition with the help of additional data, for example, accelerometer data.

Acknowledgments

This work has been conducted in the Techniques for Efficient Processing of Big Data Project in Aalto University, and was supported by The Academy of Finland and Service Innovation Research Institute (PIKE).

References


A Case Study on the Business Benefits of Automated Process Discovery

Matej Puchovsky, Claudio Di Ciccio, and Jan Mendling

Vienna University of Economics and Business
matej.puchovsky@gmail.com, {claudio.di.ciccio,jan.mendling}@wu.ac.at

Abstract. Automated process discovery represents the defining capability of process mining. By exploiting transactional data from information systems, it aims to extract valuable process knowledge. Through process mining, an important link between two disciplines – data mining and business process management – has been established. However, while methods of both data mining and process management are well-established in practice, the potential of process mining for evaluation of business operations has only been recently recognised outside academia. Our quantitative analysis of real-life event log data investigates both the performance and social dimensions of a selected core business process of an Austrian IT service company. It shows that organisations can substantially benefit from adopting automated process discovery methods to visualise, understand and evaluate their processes. This is of particular relevance in today’s world of data-driven decision making.

1 Introduction

In order to sustain competitive advantage and superior performance in rapidly changing environments, companies have intensified their efforts towards structuring their processes in a better and smarter way. Business Process Management (BPM) is considered an effective way of managing complex corporate activities. However, a consistent shift from mere process modelling and simulation towards monitoring of process execution and data exploitation can be observed nowadays [8]. Event logs, i.e., data mapping traces of process execution in modern information systems, can be used to define new process models and derive optimisation possibilities. The techniques of automated process discovery aim at extracting process knowledge from event logs and represent the initial steps of exploring capabilities of process mining.

Studying real-life enterprise data by applying process mining methods can deliver valuable insights into the actual execution of business operations and their performance. This is of particular relevance in today’s age of industries automating their processes via workflow systems [3] with growing support of process execution by various enterprise information systems. By building process definitions, models and exploring execution variations, automated process discovery has the potential to fill the information gap between business process departments and domain experts in enterprise settings. In order to assess the
business benefits of process mining, we cooperated with an Austrian IT service provider to conduct an industrial case study. The company aimed at analysing one of their core business processes using process execution data from an enterprise resource planning system (ERP). Since our industrial partner has had no previous experience with process mining, the case study focuses on a preliminary assessment of data exploitation through a process discovery initiative.

The remainder of the paper is as follows. In Section 2, we introduce the research area of process mining with its fundamental terminology and related work in the field. Section 3 describes the studied process together with the structure of the examined event log. Besides, we present the approach for our case study. The results of the empirical analysis are covered in Section 4, which is divided into the process and social views as well as a subsection covering the execution variants. Section 5 elaborates on the business impacts of the research results. Finally, Section 6 summarises our findings and gives suggestions towards future research.

2 Background

Van der Aalst [1] defines process mining as the link between data science and process science, making it possible to depict and analyse data in dynamic structures. By analysing event logs, i.e., records documenting the process executions tracked by an information system, process mining aims at retrieving process knowledge – usually in a form of process models. Depending on the setting, the extracted models may (i) define and document completely new processes, (ii) serve as a starting point for process improvements, or (iii) be used to align the recorded executions with the predefined process models. The idea to discover processes from the data of workflow management systems was introduced in [7]. Many techniques have been proposed since then: Pure algorithmic ones [6], heuristic [21], fuzzy [11], genetic (e.g., [16]), etc. ProM [2] is currently one of the most used plug-in based software environment incorporating the implementation of many process mining techniques. Process mining is nowadays a well-established field in research and examples of applications in practical scenarios can be found in various environments such as healthcare services [15], [14], financial audits [22] or public sector [5]. The organisational perspective of process mining focuses on investigating the involvement of human resources in processes. Schöning et al. [19] propose a discovery framework postulating background knowledge about the organisational structure of the company, its roles and units as a vital input for analysing the organisational perspective of processes.

3 Case study

The empirical study of the process mining methods was conducted in cooperation with an Austrian IT service provider with a diversified service portfolio. The main aim of the analysis was to evaluate the applicability of process mining for
process discovery and to provide solid fundamentals for future process improvements. Our industrial partner has a well-developed business process management department responsible for all phases of the business process management lifecycle, as defined by Dumas et al. [10]. All process models and supporting documents are saved in a centralised process management database, which is also used for storing all quality management reports.

During our investigation, we analysed the process of offering a new service (henceforth, offering process), which belongs to the core business operations in the value-chain of our industrial partner. The process describes all the necessary steps that need to be performed in order to create an offering proposal with detailed documents on a specific service, which is a subject of the offer, including pricing information for the customer. Upon approval, the offer is sent to the customer, who decides whether it meets their expectations or not. In case of an acceptance, a new contract is concluded. If the customer rejects the offer, they can either return it for adjustments, when a new version of the offering documents is created, or decline any further modifications. Up until now, almost all information regarding the process has been collected during numerous iterative sessions of interviews with the domain experts. The industrial partner selected this process due to the complex structures of its process model, which has already been simplified, as well as an extensive support through an IT system and suspected loops in process execution that are believed to negatively impact process duration and often cause lack of transparency. The primarily goal is therefore to understand how the process actually works, based on the available execution data.

3.1 Approach

Our case study aims to give answers to the following business questions:

− How does the offering process perform based on the available log data? Here, the focus is set primarily on process duration, acceptance rates of the offers, and number of iterations in the recorded cases. Moreover, we investigate what variables might have and impact on the process duration. We thus aim to exploit both the “traditional” log attributes such as timestamps or activities and “non-standard” attributes with additional process information.
− Are there any iteration loops recorded in the log, and if yes, how many? How many different variations of process execution can be identified and do they vary significantly?
− How many human resources are involved in the process execution and what roles can be identified in the event log records?

The analysis was conducted primarily by means of the process mining tool minit\textsuperscript{1}. Throughout the paper, we differentiate between the process and social view, the former dealing with the performance aspects of the process, the latter explaining its social network. The event log data were processed within both

\textsuperscript{1}http://www.minitlabs.com
frequency (i.e., absolute numbers of executions) and performance dimensions (i.e., duration metrics). Through a comparison of the process execution variants, i.e., activity sequences shared by more process instances, we aim to identify similarities between cases, but also to uncover execution anomalies recorded in the event log. The analysis of the connections between human resources and their activities in the workflows are used to cover the social perspective of the examined process.

### 3.2 Description of the event log data

The event log used in this study was created by the IT system owner via a query in the EPR system our industrial partner. The extracted cases were restricted to the workflows started in 2015. Due to the fact that no formal end activity is defined in the process, the log possibly contains several incomplete process instances. We therefore have to resort to specific attributes to determine the status of the analysed process instances. Our industrial partner provided us with a single CSV data table with 12,577 entries (activities), corresponding to 1,265 distinct process instances. Due to privacy concerns of the company, no other platforms were allowed. Therefore, the fact that we were only able to work with one data file was a major limitation for our research.

The recorded events are enriched with the following 11 attributes: `CaseID`, `Version`, `Customer`, `Segment`, `WF-Start`, `WF-Status`, `Activity`, `Start`, `End`, `Duration`, `Performer`. Overall, 258 employees were involved in the process execution. The activity names were recorded in German, therefore, we have complemented these with English translations (see Table 1) and also employ the labels in English throughout the paper.

The log attribute `workflow status` (WFS) plays an important role for our analysis, even though it is not a typical attribute such as activity name or timestamp. The values recorded under this attribute provide information about the outcome

### Table 1: English translations for the German activity labels from the event log

<table>
<thead>
<tr>
<th>German</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td>Erstellung</td>
<td>Offer creation</td>
</tr>
<tr>
<td>Fertigstellung</td>
<td>Offer completion</td>
</tr>
<tr>
<td>Fachliche Prüfung</td>
<td>Specialist check</td>
</tr>
<tr>
<td>Freigabe</td>
<td>Approval</td>
</tr>
<tr>
<td>Versand</td>
<td>Shipment</td>
</tr>
<tr>
<td>Angebotseinschaltung Kunde</td>
<td>Customer decision</td>
</tr>
<tr>
<td>Attribute vervollständigen</td>
<td>Amend attributes</td>
</tr>
<tr>
<td>AV Person festlegen</td>
<td>Appoint responsible manager</td>
</tr>
<tr>
<td>Übernahme AV</td>
<td>Takeover</td>
</tr>
<tr>
<td>SLA-Relevanz beurteilen</td>
<td>Assess SLA relevance</td>
</tr>
<tr>
<td>Abstimmung SLA Ersteller</td>
<td>Matching SLA creator</td>
</tr>
<tr>
<td>Presales – Bearbeiter dispatchen</td>
<td>Dispatch performer</td>
</tr>
<tr>
<td>Presales – Requirement Analyse</td>
<td>requirement analysis</td>
</tr>
<tr>
<td>Presales – Solution Design</td>
<td>Solution design</td>
</tr>
<tr>
<td>Presales – Ausarbeitung Angebotsinhalte</td>
<td>Drafting offering content</td>
</tr>
<tr>
<td>Presales – 4-Augen Prüfung</td>
<td>Confidentiality control</td>
</tr>
</tbody>
</table>
of the offering process or the status of the offer itself. Workflow statuses are automatically recorded at the beginning of every activity and can be changed after the activity is executed. All previous activities are then marked with the latest WFS. This attribute can also be used to determine terminal activities, which possess one of the three workflow statuses describing the outcome of an offering process – accepted, aborted or rejected.

However, a case can be assigned more than one workflow status, signalling an important relationship with another attribute – version. A good example of such case is an offer that requires adjustments after being sent to a customer, who rejected the original version. For this purpose, a new version of the offering documents is created, resulting in a new iteration of the workflow – the activities in the first iteration are recorded with new version. If the offer is accepted, the activities executed in the second loop are recorded with offer accepted. We can conclude that cases with new version contain at least one loop and that the version attribute. A closer analysis of both attributes is included in the section dealing with process performance.

4 Results

The following section describes the experimental analysis of the event log data.

4.1 Process view

In order to be able to build a process map from the event log data, the activity names and timestamps are essential. The process map is visualised as a directed graph consisting of $n$ nodes and $e$ edges. Each node represents an event or activity with an individual name and, based on the chosen dimension, the information about the execution frequency or duration.

Our initial point of interest is the process map depicting the sequence of activities under the frequency dimension, i.e. how often was a particular activity executed or in how many cases can an activity be found. In order to alter the abstraction level of the process map, we use different complexity levels of activities (nodes) and paths (edges/transitions), which determine the number of elements depicted in the model – the higher the complexity level, the more nodes/edges are visualised in the process map.

We first postulate a 0% complexity of both the activities and paths, resulting in a process model with six most frequently executed activities plus the terminal nodes start and end. In Figure 1a, the ordered activities contain the descriptive metric of case count, i.e., the frequency of cases in which the particular activity was performed. For example, it can be observed that the starting activity Offer creation was performed in 1,173 process instances of the offering workflow and is therefore accompanied by the strongest highlight around its node. Based on the recorded data, we can conclude that the model in Figure 1a represents the most common path/behaviour of the process. Increasing the percentage of activities and paths visualised in the model leads to growing complexity of the whole
Fig. 1: Process models under frequency dimension with the case frequency metric graph. This can be observed in Figure 1b, where 50% of all activities available in the event log are depicted. We can observe that additional activity types from the pre-sales phase emerge. Their presence in the model signalises preparing non-standard offers, in which a pre-sales team has to be involved to perform the requirement analysis and deliver a solution design for the desired service. The proportion of the process instances with the pre-sales phase was slightly over 40%.

The highest possible level of activity complexity uncovers a total of 18 logged process activities. At 100% activity and path complexity levels, the generated process map becomes very cluttered (see Figure 2). The graph gives us a good perception of how cross-connected the activities in the process are. The offering process evidently can be performed in numerous variations. Similarly to the starting events, the number of possible activities ending the workflow increases at higher levels of model complexity.

4.2 Variants

A variant can be defined as a set of process instances that share a specific sequence of activities. This means that all cases in a certain variant have the same start and end event and that all activities between the terminal nodes are identical and performed in the same chronological order. Variants can be very helpful for recognising different performance behaviour or irregular patterns
not conforming with the a-priori process model. In general, metrics from both frequency and performance dimensions can be assessed.

The event log of the offering process depicts a total of 280 variants with event count per case varying from 1 to 49. Due to the lack of space, we only concentrate on the top 24 performance variants that cover around 70% of all available cases.

The key goal of our variant comparison is to find similarities between the 24 investigated variants and thus build variant groups explaining the execution of the offering workflow. This allows us to identify two major groups of variants based on whether they depict offers for standard or non-standard services, let them be labelled A and B.

All variants in group A share the initial activity Offer creation and end with either Customer decision or Takeover (see Fig. 3). There are exactly nine variants contained in group A with total case coverage of almost 40% (444 cases). Since no pre-sales activities can be found in the cases of cluster A, we can conclude that all the involved process instances cover standardised offers. The presence of the activity Amend attributes signalises that iteration loops were recorded in the cases involved. The second identified variant group, B, features a higher variety of the recorded activities. It contains 15 variants with event frequency varying from 2 to 15. The cases found in this group all contain at least one activity from the pre-sales phase, thus marking process instances dealing with non-standardised services. Interestingly, no loop performance was identified in the variant group B, as the recorded cases were all missing the activity Amend attributes, i.e. a trigger for repeated process executions. Fig. 4 shows that, for both variant groups, the mean case duration tends to increase with growing number of activities per case. This can be explained both by the presence of loops or to the resource-intensive pre-sales phase.
Fig. 3: Process maps of the variant group A under frequency and performance dimension

Fig. 4: Relationship between the number of activities per case and mean case duration

4.3 Performance

For the analysed process, we define three main KPIs: (a) duration, (b) number of iterations loops in the process, and (c) process outcome. The duration metrics can be assessed by making use of the start and end timestamps, which are recorded for every activity in the log. By exploiting the data recorded under the version attribute we analyse the number of iterations in the recorded process instances. Finally, the process outcome, i.e., the acceptance rate of the offers
created through the process, is derived from the workflow statuses of the recorded cases.

Several activities stand out when it comes to the duration metrics — Customer decision accounting the mean duration of over three weeks followed by Requirement analysis with significantly lower mean duration of 1.67 weeks and Offer creation (6.03 days). These results align with the available a-priori process knowledge as they are known to require the most resources and processing time. The total duration of the analysed cases varies significantly. Therefore, we investigated the impact of three selected variables — number of activities in a case, resources involved and the number of versions a workflow possesses (since versions can be seen as iteration loops) — on the total case duration via a multivariate regression model: \( \text{case duration} = \alpha + \beta_a \cdot \text{activities} + \beta_r \cdot \text{resources} + \beta_v \cdot \text{versions} \).

The coefficients of all activities (\( \beta_a = 28.79 \)), resources (\( \beta_r = 77.883 \)) and versions (\( \beta_v = 276.154 \)) are significantly different from 0. We can therefore conclude that an increase in all selected variables results in an increased case duration with the number of versions having the highest impact. The model variables exhibit a mild degree of correlation (\( r = 0.458 \)). However, the coefficient of determination amounts to 0.21, meaning that only 21% of the case durations can be explained by the selected variables.

The version attribute denotes the version of the offering documents throughout the whole process. The document numbering is entered manually by human resources and begins with V1.0 for new cases. The version number is then automatically assigned to all new activities in the process until the version, and thus, the document, is changed (1.x for minor changes or drafts, x.0 for major alterations). Version numbers can therefore be seen from two major points of view: (i) business, where altering versions corresponds to changes in the offering documents, and (ii) process, with versions being equivalent to the number of loops in a process instance. By inducing the version number as a metric for process iteration loops, we can conclude that 1,009 (79.76%) out of the recorded 1,265 cases were executed in only one iteration. These are followed by 192 process instances (15.18%) with two iterations. Only 64 (around 5%) cases consisted of three or more loops. These findings indicate a straightforward execution of the process in the majority of examined cases.

We can also observe a relationship between the version attribute and workflow status, which is depicted in Figure 5, where the x-axis represents the number of distinct versions found in a case and the y-axis showing the number of workflow statuses per case. Generally speaking, a change in the workflow status triggers a change in the version number. We observe that most process instances are recorded in a 1:1 relationship, where the whole workflow holds only one version number and one workflow status — e.g., an offering proposal that has been accepted after the first iteration with the version number V1.0, or a proposal that has been rejected after the first iteration. The second largest group, 2:2, is common for process instances, where two distinct document versions were recorded (e.g., V1.0 and V2.0) together with two distinct workflow statuses. This behaviour is identified in offerings that were returned for adjustments by
the customer (i.e., triggering a change in the version number) and were accepted after the second iteration.

Figure 6 depicts a bar chart with all 13 workflow statuses found in the event log with the corresponding case counts. It shows the apparent dominance of Offer accepted with 715 cases (56.52% of all recorded). According to the data, only 61 offers were rejected by the customer. The sum of the cases in Figure 6 is 1,610, which is higher than the total case count in the event log (1,265). This difference can be explained by process instances that hold multiple workflow statuses. Yet, some of the recorded statuses do not provide explicit information about the process outcome. For instance, the offering process was aborted in 273 cases. Unfortunately, the data do not provide any additional information concerning the reasons for terminating the process.

Combining version numbers with the workflow status also allows for interpreting the “performance efficiency”. 70.21% (502) of all accepted offering proposals were concluded after the first iteration with the version number V1.0. However, we also discovered a significant proportion of cases that entered a new loop (242 cases or 20.23%) or were aborted (221 workflows or 28.48%) after the first iteration with the version number V1.0.
4.4 Social view

Analysing the social graph of the process helps us understand the relationships between different resources involved in the process execution. Our investigation of the social network was primarily focused on examining its frequency aspects. We resort on the investigation of relationships of the performers involved in the process execution at different complexity levels. The initial complexity levels were set to 50% for the resources and 0% for the connections, resulting in the social network consisting of 32 nodes and 48 edges depicted in Figure 7. In [20], Van Steen mentions that the key to understanding the structure of a social network is a deeper analysis of subgroups found within the network. Since the investigated social network exhibits very high complexity with 258 resources and 396 connections, this approach is also fitting for our case study. Based on the log data, the top 15 performers (5.81%) account for 60% of all the recorded events. Here, several relationships between specific performers stick out (see Table 2). Noticeable connection counts can also be observed within performers who pass the workload to themselves. The latter type of relationships appears to have significantly higher total count than the connections between resources, despite the case occurrences staying very similar. We can therefore conclude that larger sets of subsequent activities tend to be performed by the same resource, whereas handing over the workload to other employees occurs less often.

The comparison of the event and case involvement rates, i.e., respectively the share on the total number of recorded events, and the proportion of cases in which a particular resource was involved (see Table 3), provides some assumptions towards the roles of the involved employees. It can be assumed that those with a
higher share of executed events perform more activities within a process instance. On the other hand, the resources with lower event involvement rates, though active in many cases, are assumed to act as reviewers and approvers of the offering proposals who are only responsible for a few specific activities.

Considering human resources with regards to the performance aspects of the process, we set an initial hypothesis that an increasing count in activity instances per case leads to an increase in the number of employees involved in the process execution. However, in process instances containing loops, the activities within the new iterations are assumed to be performed by the same resources who have already been employed in the execution of previous iterations and know the case specifications. This is reflected in lower growth rate of distinct human resources per case.

The scatter plot in Figure 8 supports our hypothesis. Since the data exhibit a rather quick initial increase that then levels out, we computed a logarithmic trendline for the development of “resource population” per case with growing activity counts. With the available process knowledge, the log-trendline can be interpreted as follows: The initial quick increase in resource counts per case is linked to the difference between cases with standard and non-standard services, where the resource-consuming pre-sales phase is required. As the employees of the pre-sales team differ from those who actually create the offering documents, the differences between these two groups are higher. Subsequently, this results in a quick increase of resource counts. However, further increase in activity count per case leads to a slower growth rate in the resource count. These findings support our hypothesis that with multiple iterations in the process, the repeated activities are executed by the same human resources who have already been involved in previous iterations and the “intake” rate of new employees levels out. The level-log regression equation, \( \text{resources} = -0.7005 + 2.6545 \ln(\text{activities}) \), shows that an increase in the activity count per case by one percent, we expect the resource count to increase by \( (2.6545/100) \) human resources. The coefficient of determination of 0.7372 indicates a good fit of the regression model with a significant result \( (p < 0.001) \).
5 Discussion

Our study evidenced that applying methods of automated process discovery can yield useful insights into both the performance and the social structure of business processes. By analysing event log data of an ERP system, we detected a significant number of differing execution variants among the recorded cases of the investigated process. We have shown that duration can be influenced by the number of activities per case, resources and number of loops. The presence of a pre-sales phase in offering workflows covering non-standard services often causes a longer case duration and higher numbers of employees involved in the process. Therefore, we propose the portfolio with standard services of our industrial partner to be extended by new services that have previously successfully met customer expectations. This could contribute to reducing the number of cases with the pre-sales phase and the execution time of the process.

From the offering proposals recorded in the investigated log, around 56.5% were accepted, a figure that our industrial partner aims to improve in the future. On the other hand, over 70% of the contracts were concluded after only one process iteration, which is a positive result. Considering the presence of loops in the process, the investigated cases were performed in a relatively straight-forward way with almost 80% of cases finished without any loops. Therefore, the initial suspicion that the majority of cases included several iteration loops was not confirmed. Still, it is proposed to introduce a comprehensive list of all customer change requests once the offering documents are returned for adjustments (and thus triggering new loop). That way, future performance of the process in the first iteration can be improved.

Based on the events analysis, the case counts and the social network diagram, the roles of the recorded resources were identified. Interestingly, the very high number of employees involved in the process execution was in fact unknown to our industrial partner. Even though 258 employees were involved in the examined cases, most of the activities were performed by only few individuals. Therefore, it is advised to assess whether to distribute the workload more evenly among the
available employees or to reduce the resource counts for this particular process to only the key personnel.

Our research was limited by the restriction on tools that we were allowed to use by the industrial partner and by the lack of information about the specifications of the IT system used as the data source. Nonetheless, it can be concluded that these results represent a solid basis for future decisions with regards to process optimisation.

6 Conclusion

In this paper, we demonstrated how process mining can be used to extract valuable knowledge about business processes from transactional data. Our study provides evidence that by exploiting all attributes of the event log, it is possible to obtain extensive knowledge about the dynamics, performance and the resource structure of the examined process.

Our investigation was hampered by a relatively high number of events for which the resource was not specified. Such events were also associated to missing end-timestamps. Therefore, we chose not to include those events in the analysis of both the process variants and the social network, as they would have yielded biased results otherwise. The lack of reported information often affects real-world logs. It is therefore in the future plans to integrate the techniques of Rogge-Solti et al. [17, 18] to replace missing values with estimates, so as not to disregard events which may otherwise be relevant because of other information they bring. Considering the social perspective, we plan to acquire further knowledge about the organisational structure of the enterprises in future case studies as well as the role allocation of the available individuals involved in the process execution. This is of particular importance when verifying whether the designated responsibilities set for the execution of specific activities by the enterprise are not violated.

In the light of the promising results achieved, it is in our plans to collaborate further with our industrial partner to deepen the investigation by means of other process mining techniques, in the spectrum both of process discovery and conformance checking. Furthermore, the high variability of recorded process instance executions could also suggest that clarifying results might derive from declarative process discovery [13, 9, 12]. The declarative approach indeed aims at understanding the behavioural constraints among activities that are never violated during the process enactment, rather than specifying all variants as branches of an imperative model. As a consequence, it is naturally suited to flexible workflows [4].

References

Model and Event Log Reductions to Boost the Computation of Alignments

Farbod Taymouri and Josep Carmona

Universitat Politècnica de Catalunya, Barcelona (Spain)
{taymouri,jcarmona}@cs.upc.edu

Abstract. The alignment of observed and modeled behavior is a pivotal issue in process mining because it opens the door for assessing the quality of a process model, as well as the usage of the model as a precise predictor for the execution of a process. This paper presents a novel technique for reduction of a process model based on the notion of indication, by which, the occurrence of an event in the model reveals the occurrence of some other events, hence relegating the later set as less important information when model and log alignment is computed. Once indications relations are computed in the model, both model and log can be reduced accordingly, and then fed to the state of the art approaches for computing alignments. Finally, the (macro)-alignment derived is expanded in these parts containing high-level events that represent a set of indicated events, by using an efficient algorithm taken from bioinformatics that guarantees optimality in the local parts of the alignment. The implementation of the presented techniques shows a significant reduction both in computation time and in memory usage, the latter being a significant barrier to apply the alignment technology on large instances.

1 Introduction

Nowadays many systems generate event logs, which are footprints left by process executions. Process mining delves into this information, and examines it to extract, analyze and enhance evidence-based process models [10]. One of the challenges in process mining is how to align a process model to a set of traces forming an event log. Given a trace representing a real process execution, an optimal alignment provides the best trace the process model can provide to imitate the observed trace [1]. Alignments are crucial for important metrics like fitness, precision and generalization [1,2].

This paper presents a model-based technique for reduction of a process model and observed behavior that both preserves the semantics of the process model and retains the information of the original observed behavior as much as possible. The technique is meant to fight the main problem current approaches for alignment computation have: the complexity both in space and time. The overall idea relies on the notion of indication between activities of the process model when it is represented as a Petri net. An indication relation between a set of transitions (indicated set) and another transition (indicator) denotes a causal
firing relation in the model, which expresses that the presence in any model's sequence of the indicator transition requires the presence of the indicated set as well. The notion of indication is inspired from the reveals relation from [3]. We use a well-known technique to find logically independent parts of a graph (known as SESEs in [7]), which are then used to gather indication relations efficiently. These relations dictate which parts of a process model are abstracted as a single, high-level node. Once the model is reduced, the observed trace to align is projected (hence, reduced as well) into the reduced model's alphabet. This way, not only the model but also the trace are reduced, which in turn makes the alignment techniques to be significantly alleviated, specially for well-structured process models where many indication relations may exist. Once alignments are computed, the final step is also an interesting contribution of this paper: to cast the well-known Needleman–Wunsch algorithm [6] to expand locally each high-level part of the alignment computed, using the indication relation.

2 Related Work

The seminal work in [1] proposed the notion of alignment, and developed a technique to compute optimal alignments for a particular class of process models. For each trace \( \sigma \) in the log, the approach consists on exploring the synchronous product of model’s state space and \( \sigma \). In the exploration, the shortest path is computed using the \( A^* \) algorithm, once costs for model and log moves are defined. The approach is implemented in ProM, and can be considered as the state-of-the-art technique for computing alignments. Several optimizations have been proposed to the basic approach to speed up and improve memory consumption. The recent work in [9] proposed a divide and conquer strategy based on Integer Linear Programming (ILP) approach to compute approximate alignments. Despite its memory and time efficiency, it cannot guarantee the obtention of an (optimal) alignment.

The work in [5] presented a decomposition approach using SESEs for conformance checking of the model and observed behavior. The proposed approach decomposes a given model to smaller parts via SESE and then applies conformance checking for each part independently. This technique is very efficient, but the result is decisional (a yes/no answer on the fitness of the trace). Recently [12] proposed a new approach which provides an algorithm that is able to obtain such an optimal alignment from the decomposed alignments if this is possible, which is called proper optimal alignment. Otherwise, it produces a so-called pseudo-alignment which as in the case of [9], may not be executable in the net.

The seminal work [4] first introduced the notion of reveals relation, which determines that whenever an action \( a \) happens, then the occurrence of another action \( b \) is inevitable. The notion of indication of this paper is inspired on the reveals relation.

The Refined Process Structure Tree (RPST), proposed by [11], is a graph parsing technique that provides well-structured parts of a graph. The resulting parse tree is unique and modular, i.e., local change in the local workflow graph
results in a local change of the parse tree. It can be computed in linear time using the method proposed in [8] which is based on the triconnected components of a given biconnected graph. The proposed approach only works with single sink, single source workflow graphs which hampers its applicability to real world problems with many sink, source nodes. The work in [7] presents a more efficient way to compute RPST which can deal with multiple source, sink workflow graphs.

3 Preliminaries

A Petri Net is a 3-tuple \(N = (P, T, F)\), where \(P\) is the set of places, \(T\) is the set of transitions, \(P \cap T = \emptyset\), \(F : (P \times T) \cup (T \times P) \rightarrow \{0, 1\}\) is the flow relation. Marking of a Petri net represents the number of tokens each place has. Given a node \(x \in P \cup T\), its pre-set and post-set (in graph adjacency terms) are denoted by \(\bullet x\) and \(x \bullet\) respectively. WF-net is a Petri net where there is a place \(\text{start}\) (denoting the initial state of the system) with no incoming arcs and a place \(\text{end}\) (denoting the final state of the system) with no outgoing arcs, and every other node is within a path between \(\text{start}\) and \(\text{end}\). Fig. 1(a) represents a WF-net. Given an alphabet of events \(T = \{t_1, \ldots, t_n\}\), a trace is a word \(\sigma \in T^*\) that represents a finite sequence of events. An event log \(L \in \mathcal{B}(T^*)\) is a multiset of traces\(^1\). An alignment is represented by a two-row matrix where the top and bottom rows represent moves on log and the model respectively. For example given trace \(t_1 t_4 t_2 t_5 t_8\) and the model in Fig. 1(a), an example of alignment is:

\[
\alpha = \begin{bmatrix}
\downarrow t_1 & \downarrow t_4 & \downarrow t_2 & \downarrow t_5 & \downarrow t_8 \\
\uparrow t_1 & \uparrow t_2 & \uparrow t_4 & \uparrow t_5 & \uparrow t_8
\end{bmatrix}
\]

Let \(F \subseteq E\) represents a set of edges of a directed graph \(\langle V, E, \ell \rangle\), \(G_F = \langle V_F, F \rangle\) is the subgraph formed by \(F\) if \(V_F\) is the smallest set of nodes such that \(G_F\) is a subgraph. A node in \(V_F\) is boundary with respect to \(G_F\) if it is connected to nodes in \(V_F\) and in \(V - V_F\), otherwise it is interior. A boundary node \(u\) of \(G_F\) is an entry node if no incoming edge of \(u\) belongs to \(F\) or if all outgoing edges of \(u\) belong to \(F\). A boundary node \(v\) of \(G_F\) is an exit node of \(G_F\) if no outgoing edge of \(v\) belongs to \(F\) or if all incoming edges of \(v\) belong to \(F\). \(G_F\) with one entry and one exit node is called SESE. If a SESE contains only one edge it is called trivial. A SESE of \(G\) is called canonical if it does not overlap with any other SESEs of \(G\), but it can be nested or disjoint with other SESEs. For example in Fig. 1(b) all SESEs are canonical, \(S_2\) and \(S_4\) are nested, \(S_3\) and \(S_2\) are disjoint. A WF-net can be viewed as a Workflow graph if no distinctions are made between its nodes. WF-graph of Fig. 1(a) is presented in Fig. 1(b). Let \(G\) be a graph, then its Refined Process Structure Tree (RPST) is the set of all canonical SESEs of \(G\). Because canonical fragments are either nested or disjoint, they form a hierarchy. In a typical RPST, the leaves are trivial SESE and the root is the whole graph. Fig. 1(c) is the RPST of WF-graph in Fig. 1(b),

\(^1\mathcal{B}(A)\) denotes the set of all multisets of the set \(A\).
$S_1$ which is the entire graph is at root and leaves are trivial SESEs which only contain one edge.

Fig. 1: (a) WF-net, (b) Workflow graph, (c) RPST, (d) Reduced WF-net

4 Overall Framework

Given a process model $N$, represented by a Petri net, and $\sigma$ as observed behavior, the strategy of this paper is sketched in Fig. 2. We now provide descriptions of each stage.

Fig. 2: Overall framework for boosting the computation of alignments
– **Model Reduction**: \( N \) will be reduced based on the notion of indication relation which results in \( N_r \). It contains some abstract events representing the indicators of certain indicated sets of transitions. Section 5.1 explains it in detail.

– **Log Reduction**: Using the indication relations computed in the model, \( \sigma \) is projected into the remaining labels in \( N_r \), resulting in \( \sigma_r \). Section 5.2 describes this step.

– **Computing Alignment**: Given \( N_r \) and \( \sigma_r \), approaches like [1] and [9] can be applied to compute alignments. At this point because both \( N_r \) and \( \sigma_r \) contain abstract events, the computed alignment will have them as well. We call it **macro-alignment**.

– **Alignment Expansion**: For each abstract element of a macro-alignment, the modeled and observed indications are confronted. Needleman–Wunsch algorithm [6] is adapted to compute optimal alignments. Section 6 will be centered on this.

5 **Reduction of Model and Observed Behavior**

5.1 **The Indication Relation**

Let us consider the model in Fig. 1(a). For any sequence of the model, whenever transition \( t_4 \) fires it is clear that transitions \( t_1 \), \( t_3 \), and \( t_2 \) have fired as well.

Formally:

**Definition 1 (Indication Relation)**. Let \( N = \langle P, T, F \rangle \), \( \forall t \in T \), indication is defined as a function, \( I(t) \) where, \( I : T \rightarrow [P(T)^{+}]^{+} \) \( I(t) \) such that for any sequence \( \sigma \in \mathcal{L}(N) \), if \( t \in \sigma \) then \( I(T) \in \sigma \). If \( I(t) = \omega_1 \omega_2 \ldots \omega_n \), then elements of \( \omega_m \) precede the elements of \( \omega_n \) in \( \sigma \) for \( 1 \leq m < n \). \( I(t) \) is called linear if it contains only singleton sets, i.e. \( \forall \omega_i \in I(t), |\omega_i| = 1 \) otherwise it is non-linear.

For example in Fig. 1(a), \( I(t_4) = \{t_1\}\{t_2\}\{t_3\}t_4 \) (non-linear) and \( I(t_8) = \{t_7\}t_8 \) (linear). SESEs are potential candidates for identifying indication relations inside a WF-net: the exit node of a SESE is the potential indicator of the nodes inside the SESE. Since entry/exit nodes of a SESE can be either place or transitions, SESEs are categorized as Fig. 3: Linear SESEs and their reduction. \( (P, P) \), \( (P, T) \), \( (T, P) \) or \( (T, T) \). In case the SESE is linear, indication relations can be extracted easily and the corresponding SESE is reduced (see Fig.3).

Non-linear cases are decomposed into linear ones so that indication relations can be computed directly on the linear components extracted. After that, the indication relation of the corresponding linear SESEs are computed and they are
reduced as well. This procedure should be done with caution to avoid reaching a deadlock situation. Hence a post verification must be done after reduction of these linear parts. Informally, the verification is only needed for particular type of linear SESEs \((T, T)\), and consists on validating the property of the SESE after the reduction. Notice the verification is necessary in these cases because, non-linear SESEs may contain linear indications at nested level, but which cannot be extracted due to choice or loop constructs (see See Fig. 4).

Fig. 4: (a) Non-Linear \((T, T)\), (b) Non-Linear \((P, T)\). Here indication relations cannot be computed at the whole SESE level, i.e., in (a) \(t_5\) does not indicate neither \(t_2\) nor \(t_4\).

The reduction schema is depicted in Fig. 5. From the RPST, a top-down approach is applied that searches for indication-based reductions that do preserve the language of the initial model, once the net is expanded back, i.e., the language of the model must be preserved after reduction. As mentioned above, the reduction of non-linear SESEs must be done alongside by a post verification; for instance, Fig. 6 shows that in spite of the indication arising from SESE \(S_2\), the net cannot be reduced without changing the language. To put it another way, this reduction will cause a deadlock in the reduced model, and hence must be avoided. Looking at the reduced result in Fig. 6, transition \(t_5\) (New) will never fire because after the reduction. Notice that the reduction can be applied more than once till saturation (hence the arc back from the node “Reduced WF-net” to the node “WF-net”).

Fig. 5: Schema for reduction of a WF-net.

Fig. 7 shows an example (for the sake of simplicity only linear SESEs are shown). Obviously, SESE \(S_2\) is inherently a linear SESE but the rest come from the decomposition of non-linear SESEs. The reduction schema is as follows: Since \(S_2\) is inherently a linear SESE, hence it can be reduced easily according to Fig. 3 without any post verification. The rest of linear SESEs also will be reduced.
Fig. 6: Incorrect indication-based reduction: a deadlock is introduced.

accordingly and post verification will be done after each reduction to check that no deadlock arises. One can see all reductions will be pass the verification, except for $S_7$, whose reduction induces a deadlock. Applying the reduction once, results in Fig. 7(b). As mentioned earlier, the reduction can be applied more than once until no reduction can be made. Fig. 7(c) is the reduction of the model in Fig. 7(b) and it is clear that no more reduction can be made from this model.

### 5.2 Reduction of Observed Behavior

Given a reduced model $N_r$ and $\sigma$, we show how to produce $\sigma_r$. We will use the reduced model in Fig. 7(b) and the trace $\sigma_1 = t_1t_3t_1t_10t_21t_6t_2t_7t_16t_25t_19t_20t_26$. The indication of $t_{(New)}$ in Fig. 7(b) which is linear, equals to $\{t_5\}$ of $t_{15}$. So the observed indication for this abstract node is $\sigma_{1\downarrow I(t_{(New)})} = t_5$. After computing the observed indication the reduced trace is $t_1t_5(t_{New})t_3t_1t_10t_21t_6t_2t_7t_16t_25t_19t_20t_26$. For $t_{17(\text{New})}$, $I(t_{17(\text{New})}) = \{t_3\}\{t_11\}\{t_{17}\}$, which is non-linear and merged of two linear indications, $I_1(t_{17(\text{New})}) = \{t_3\}\{t_{10}\}\{t_{17}\}$ and $I_2(t_{17(\text{New})}) = \{t_3\}\{t_{11}\}\{t_{17}\}$. So the projection must be done for each linear indication separately, $\sigma_{1\downarrow I_1(t_{17(\text{New})})} = t_3t_{10}$ and $\sigma_{1\downarrow I_2(t_{17(\text{New})})} = t_3t_{11}$, removing transitions $t_3$, $t_{10}$, $t_{11}$ and $t_{17}$ from the current trace (notice that $t_{17}$ does not appear originally, hence it is not projected). Finally, we need to insert $t_{17(\text{New})}$ into the reduced trace; it will be inserted at the position of $t_{10}$, because the end transition of the abstract node, i.e. $t_{17}$ did not happened in $\sigma$, and $t_{10}$ happened last in $\sigma$.

Therefore the reduced trace so far is $t_1t_{5(\text{new})}t_{17(\text{new})}t_{21}t_6t_2t_7t_{16}t_{25}t_{19}t_{20}t_{26}$. By applying this process for the rest of abstract nodes $(t_{16(\text{New})}, t_{22(\text{New})})$, we reach $\sigma_r = t_1t_{5(\text{new})}t_{17(\text{new})}t_{21}t_{16(\text{New})}t_{22(\text{New})}t_{26}$. 

56
After reducing a given process model and corresponding observed behavior, we can use current methods for computing alignments \([1,9]\) to align \(N_r\) and \(\sigma_r\), deriving \(\alpha_r\). For example the following is the macro alignment of \(\sigma_{1r} = t_1 t_5^{(new)} t_1^{(new)} t_21 t_16^{(New)} t_22^{(New)} t_26\) and the model in Fig. 7(b).

\[
\alpha_r = \begin{array}{cccc}
  t_1 & t_5^{(new)} & t_1^{(new)} & t_21 \\
  & t_17^{(New)} & t_21 & t_16^{(New)} \\
  t_1 & t_22^{(New)} & t_26 & t_21
\end{array}
\]
When mapped to linear indications, indication of an abstract node and the corresponding observed indication are both sequence of events; hence for each linear combination of modeled/observed indication, we can adapt the dynamic programming approach from [6] (used in bioinformatics) to align two sequences. As example, we use indication of $t_{17}^{(New)}$ and its observed indication computed in the previous section.

To achieve this goal, we create a table for each linear indication, where the first row and column are filled by observed and abstract node indications respectively, as depicted in Table 1(a), 1(b). The second row and second column are initialized with numbers starting from 0,-1,-2,..., they are depicted in yellow color. The task then is to fill the remaining cells as follows:

\[
\text{SIM}(t_i, t_j) = \max(\text{SIM}(t_{i-1}, t_{j-1}) + s(t_i, t_j), \text{SIM}(t_{i-1}, t_j), \text{SIM}(t_i, t_{j-1}) - 1)
\]

Where $\text{SIM}(t_i, t_j)$ represents the similarity score between $t_i$ and $t_j$, $s(t_i, t_j)$ is the substitution score for aligning $t_i$ and $t_j$, it is 0 when they are equal and otherwise.

The final step in the algorithm is the trace back for the best alignment. In the above mentioned example, one can see the bottom right hand corner in for example Table 1, score as -1. The important point to be noted here is that there may be two or more alignments possible between the two example sequences. The current cell with value -1 has immediate predecessor, where the maximum score obtained is diagonally located and its value is 0. If there are two or more values which points back, suggests that there can be two or more possible alignments. By continuing the trace back step by the above defined method, one would reach to the 0th row, 0th column. Following the above described steps, alignment of two sequences can be found.

Alignments can be represented by a sequence of paired elements, for example $\alpha_1 = (t_3, t_3)(t_{11}, t_{11})(\perp, t_{17})$, $\alpha_2 = (t_3, t_3)(t_{10}, t_{10})(\perp, t_{17})$ and final alignment which represent the non-linear indication is $\alpha = (t_3, t_3)\{(t_{11}, t_{11}), (t_{10}, t_{10})\}(\perp, t_{17})$. This information is booked for each abstract node.

After computing local alignments for abstract nodes, we can use them to expand corresponding abstract nodes in a given $\alpha_r$. The policy of expansion depends on whether the abstract node is in synchronous or asynchronous move.

In $\alpha_r$, $t_{17}^{(New)}$ is in a synchronous move so we can expand it by its local alignment, which results in:

\[
\alpha = [t_1, t_5^{(New)}, t_3, t_11, t_{10}, \perp, t_{21}, \perp, \perp, t_{16}^{(New)}, t_{22}^{(New)}, t_{26}^{(New)}]
\]
The same story also happens for $t_{16(\text{New})}$ and $t_{22(\text{New})}$, which results in:

$$\alpha = \left[ \begin{array}{cccccccccccc}
\text{f}_1 & \text{f}_5(\text{New}) & \text{f}_3 & \text{f}_{11} & \text{f}_{10} & \downarrow & \downarrow & \downarrow & \text{f}_6 & \text{f}_2 & \text{f}_5 & \downarrow & \downarrow & \text{f}_{16} & \text{f}_{25} & \text{f}_{19} & \text{f}_{20} & \downarrow & \text{f}_{26}
\end{array} \right]$$

On the other hand $t_{5(\text{New})}$ in $\alpha_r$ is an asynchronous move both on the model and observed trace. The policy of expansion is to expand move on log and move on model independently. To put it in another way, move on log will be expanded using observed indication and move on model will be expanded using the abstract node indication, which results:

$$\alpha = \left[ \begin{array}{cccccccccccc}
\text{f}_1 & \text{f}_5 & \text{f}_3 & \text{f}_{11} & \text{f}_{10} & \downarrow & \downarrow & \downarrow & \text{f}_6 & \text{f}_2 & \text{f}_5 & \downarrow & \downarrow & \text{f}_{16} & \text{f}_{25} & \text{f}_{19} & \text{f}_{20} & \downarrow & \text{f}_{26}
\end{array} \right]$$

7 Experiments

The technique presented in this paper has been implemented in Python as a prototype tool. The tool has been evaluated over different family of examples, alongside with the state of the art techniques for computing alignments [9] ($ILP.R$), [1] ($A^*$). We used benchmark datasets from [9], [5], and a new dataset.

**Reduction of Models.** Table 2 provides the results of one-time reduction by applying the proposed method to benchmark datasets. Significant reductions are found often. Obviously one can see that the results of reduction is more representative for models without loops or contain small loops, like (Banktransfer).

**Quality of Alignments.** Since the alignment technique $ILP.R$ may be approximate, Table 3 provides an overview of how many of the computed alignments...
can be replayed for ILP.R method when combined with the technique of this paper. Comparing with Original Alignments. Table 4 reports the evaluation of the quality of the results for both approaches [1], [9] with and without applying the technique of this paper. Columns ED/Jaccard report the edit/Jaccard distance between the sequences computed, while MSE columns reports the mean square root error between the corresponding fitness values. Edit distances are often large, but interestingly this has no impact on the fitness, since when expanding abstract nodes although the final position may differ, the model still can replay the obtained sequences very often.

Memory Usage. The memory usage of computing alignments using [1], is reduced significantly. For large models, prDm6, prFm6, prGm6, M6, M10, it can only compute alignments if applied in combination of the technique of this paper. Computation Time Comparison. Fig. 8(a)-(b) report computation times for BPM-2013 and other benchmark datasets respectively. It is evident that A* approach combined with the proposed method is significantly faster than the other approach in nearly all datasets except (prGm6, prDm6, M6, M10). Still A* approach cannot compute alignments for models M6 and M10 even after applying the presented technique, and in that case the combination of ILP.R with the presented technique is the best choice.

8 Conclusion and Future Work

We have presented a technique that can be used to significantly alleviate the complexity of computing alignments. The technique uses the indication relation

\[ A^* \]
Fig. 8: Computation time for (a) BPM2013 and (b) synthetic datasets.

References


DB-XES: Enabling Process Discovery in the Large

Alifah Syamsiyah, Boudewijn F. van Dongen, Wil M.P. van der Aalst

Eindhoven University of Technology, Eindhoven, the Netherlands
A.Syamsiyah@tue.nl, B.F.v.Dongen@tue.nl, W.M.P.v.d.Aalst@tue.nl

Abstract. Dealing with the abundance of event data is one of the main process discovery challenges. Current process discovery techniques are able to efficiently handle imported event log files that fit in the computer’s memory. Once data files get bigger, scalability quickly drops since the speed required to access the data becomes a limiting factor. This paper proposes a new technique based on relational database technology as a solution for scalable process discovery. A relational database is used both for storing event data (i.e. we move the location of the data) and for pre-processing the event data (i.e. we move some computations from analysis-time to insertion-time). To this end, we first introduce DB-XES as a database schema which resembles the standard XES structure, we provide a transparent way to access event data stored in DB-XES, and we show how this greatly improves on the memory requirements of a state-of-the-art process discovery technique. Secondly, we show how to move the computation of intermediate data structures, such as the directly follows relation, to the database engine, to reduce the time required during process discovery. The work presented in this paper is implemented in ProM tool, and a range of experiments demonstrates the feasibility of our approach.

Keywords: process discovery, process mining, big event data, relational database

1 Introduction

Process mining is a research discipline that sits between machine learning and data mining on the one hand and process modeling and analysis on the other hand. The goal of process mining is to turn event data into insights and actions in order to improve processes [15]. One of the main perspectives offered by process mining is process discovery, a technique that takes an event log and produces a model without using any a-priori information. Given the abundance of event data, the challenge is to enable process mining in the large. Any sampling technique would lead to statistically valid results on mainstream behavior, but would not lead to insights into the exceptional behavior, which is typically the goal of process mining.

In the traditional setting of process discovery, event data is read from an event log file and a process model describing the recorded behavior is produced,
as depicted in Figure 1(a). In between, there is a so-called intermediate structure, which is an abstraction of event data in a structured way, e.g. the directly follows relation, a prefix-automaton, etc. To build such an intermediate structure, process mining tools load the event log in memory and build the intermediate structure in the tool, hence the analysis is bound by the memory needed to store both the event log and the immediate structure in memory. Furthermore, the time needed for the analysis includes the time needed to convert the log to the intermediate structure.

To increase the scalability, relational databases have been proposed for storing event data [17], as depicted in Figure 1(b), i.e. the event log file is replaced by a database. In [17] a database schema was introduced to store event data and experiments showed the reduction in memory use. A connection is established from the database to process mining tools to access the event data on demand using the standard interfaces for dealing with event logs, i.e. OpenXES [6]. Since no longer the entire event log is to be read in memory, the memory consumption of the process mining analysis will be shown to be reduced significantly as now only the intermediate structure needs to be stored. However, this memory reduction comes at a cost of analysis time since access to the database is several
orders of magnitude slower than access to an in-memory event log while building
the intermediate structure for further analysis.

Therefore, we present a third solution, called DB-XES, where we not only
move the location of the event data, but also the location of such intermediate
structures. In order to do so, we move the computation of intermediate structures
from analysis time to insertion time, as depicted in Figure 1(c). In other words,
each intermediate structure is kept up-to-date for each insertion of a new event
of a trace in the database. In this paper we present the general idea and a con-
crete instantiation using the intermediate structure of a state-of-the-art process
discovery technique. We show that the proposed solution saves both memory
and time during process analysis.

The remainder of this paper is organized as follows. In Section 2, we discuss
some related work. In Section 3, we present the database schema for DB-XES.
In Section 4, we extend DB-XES with the notion of intermediate structure. In
Section 5, we show how a well-known intermediate structure can be computed
inside the database. Then, in Section 6, we present experiments using the In-
ductive Miner. These show significant performance gains. Finally, we conclude
and discuss the future work in Section 7.

2 Related Work

One of the first tools to extract event data from a database was XESame [20]. In
XESame users can interactively select data from the database and then match it
with XES elements. However, the database is only considered as a storage place
of data as no direct access to the database is provided.

Similar to XESame, in [3] a technique is presented where data stored in
databases is serialized into an XES file. The data is accessed with the help of
two ontologies, namely a domain ontology and an event ontology. Besides that,
the work also provided on-demand access to the data in the database using query
unfolding and rewriting techniques in Ontology Based Data Access [9]. However,
the performance issues make this approach unsuitable for large databases.

Some commercial tools, such as Celonis1 and Minit2, also incorporate features
to extract event data from a database. The extraction can be done extremely
fast, however, its architecture has several downsides. First, it is not generic since
it requires a transformation to a very specific schema, e.g. a table containing in-
formation about case identifier, activity name, and timestamp. Second, it cannot
handle huge event data which exceed computer’s memory due to the fact that
the transformation is done inside the memory. Moreover, since no direct access
to the database is provided, some updates in the database will lead to restarting
of the whole process in order to get the desired model.

Building on the idea of direct access to the database, in [17], RXES was
introduced before as the relational representation of XES and it was shown that
RXES uses less memory compared to the file-based OpenXES and MapDB XES

1 http://www.celonis.de/en/
2 http://www.minitlabs.com/
Lite implementations. However, its application to a real process mining algorithm was not investigated and the time-performance analysis was not included.

In [21], the performance of multidimensional process mining (MPM) is improved using relational databases techniques. It presented the underlying relational concepts of PMCube, a data-warehouse-based approach for MPM. It introduced generic query patterns which map OLAP queries to SQL to push the operations to the database management systems. This way, MPM may benefit from the comprehensive optimization techniques provided by state-of-the-art database management systems. The experiments reported in the paper showed that PMCube provides a significantly better performance than PMC, the state-of-the-art implementation of the Process Cubes approach.

The use of database in process mining gives significance not only to the procedural process mining, but also declarative process mining. The work in [11] introduced an SQL-based declarative process mining approach that analyses event log data stored in relational databases. It deals with existing issues in declarative process mining, namely the performance issues and expressiveness limitation to a specific set of constraints. By leveraging database performance technology, the mining procedure in SQLMiner can be done fast. Furthermore, SQL queries provide flexibility in writing constraints and it can be customized easily to cover process perspective beyond control flow.

Apart from using databases, some other techniques for handling big data in process mining have been proposed [2, 10, 12], two of them are decomposing event logs [1] and streaming process mining [7, 18]. In decomposition, a large process mining problem is broken down into smaller problems focusing on a restricted set of activities. Process mining techniques are applied separately in each small problem which then they are combined to get an overall result. This approach deals with exponential complexity in the number of activities of most process mining algorithms [13]. Whereas in streaming process mining, it provides online-fashioned process mining where the event data is freshly produced, i.e. it does not restrict to only process the historical data as in traditional process mining. Both approaches however require severe changes to the algorithms used for analysis and they are therefore not directly applicable to existing process mining techniques.

3 DB-XES as Event Data Storage

In the field of process mining, event logs are typically considered to be structured according to the XES standard [6]. Based on this standard, we create a relational representation for event logs, which we called DB-XES. We select relational databases rather than any other type of databases, e.g. NoSQL [19], because of the need to be able to slice and dice data in different ways. An e-commerce system, for example, may need to be analyzed using many views. One view can be defined based on customer order, other view may also be defined based on delivery, etc. Some NoSQL databases, such as key-value store databases, document databases, or column-oriented databases, are suitable for the data
which can be aggregated, but have difficulties supporting multiple perspectives at the same time. Besides, relational databases are more mature than NoSQL databases with respect to database features, such as trigger operations.

Figure 2 shows the basic database schema of DB-XES. The XES main elements are represented in tables log, trace, event, and attribute. The relation between these elements are stored in tables log_has_trace and trace_has_event. Furthermore, classifier and extension information related to a log can be accessed through tables log_has_classifier and log_has_extension. Global attributes are maintained in the table log_has_global. In order to store the source of event data, we introduce the event_collection table.

OpenXES is a Java-based reference implementation of the XES standard for storing and managing event log data [6]. OpenXES is a collection of interfaces and corresponding implementations tailored towards accessing XES files. In consequence of moving event data from XES files to DB-XES, we need to implement some Java classes in OpenXES. Having the new version of OpenXES, it allows for any process mining techniques capable of handling OpenXES data to be used on DB-XES data. The implementation is distributed within the DBXes package in ProM (https://svn.win.tue.nl/repos/prom/Packages/DBXes/Trunk/).

The general idea is to create SQL queries to get the event data for instantiating the Java objects. Access to the event data in the database is defined for each element of XES, therefore we provide on demand access. We define a log, a trace, and an event based on a string identifier and an instance of class
Connection in Java. The identifier is retrieved from a value under column id in log, trace, and event table respectively. Whereas the instance of class Connection should refer to the database where we store the event data. Upon initialization of the database connection, the list of available identifiers is retrieved from the database and stored in memory using global variables.

4 Extending DB-XES with Intermediate Structures

In the analysis, process mining rarely uses event data itself, rather it processes an abstraction of event data called an intermediate structure. This section discusses the extension of DB-XES with intermediate structures. First, we briefly explain about several types of intermediate structures in process mining, then we present a highly used intermediate structure we implemented in DB-XES as an example.

There are many existing intermediate structures in process mining, such as the eventually follows relation, no co-occurrence relation, handover of work relation, and prefix-closed languages in region theory. Each intermediate structure has its own functions and characteristics. Some intermediate structures are robust to filtering, hence we may get different views on the processes by filtering the event data without recalculation of the intermediate structure like eventually follows relation, but some require full recomputation. Mostly intermediate structures can be computed by reading the event data in a single pass over the events, but some are more complex to be computed. In general the size of intermediate structure is much smaller than the size of the log, but some intermediate structures are bigger than the log.

– The directly follows relation \((a > b)\) contains information that \(a\) is directly followed by \(b\) in the context of a trace. This relation is not robust to filtering. Once filtering happens, the relation must be recalculated. Suppose that \(a\) is directly followed by \(b\), i.e. \(a > b\), and \(b\) is directly followed by \(c\), i.e. \(b > c\). If we filter \(b\), now \(a\) is directly followed by \(c\), hence a new relation \(a > c\) holds.

– The eventually follows relation \((V(a, b))\) is the transitive closure of the directly follows relation: \(a\) is followed by \(b\) somewhere in the trace. Suppose that \(a\) is eventually followed by \(b\), i.e. \(V(a, b)\), and \(a\) is eventually followed by \(c\), i.e. \(V(a, c)\). If we filter \(b\), \(a\) is still followed by \(c\) somewhere in the trace, i.e. \(V(a, c)\) still holds. Therefore, eventually follows relation is robust to filtering.

– The no co-occurrence relation \((R(a, b))\) counts the occurrences of \(a\) with no co-occurring \(b\) in the trace. For example, \(a\) occurs four times with no co-occurring \(b\), i.e. \(R(a, b) = 4\), and \(a\) occurs three times with no co-occurring \(c\), i.e. \(R(a, c) = 3\). If we filter \(b\), it does not effect the occurrence of \(a\) with no \(c\), i.e. \(R(a, c) = 3\) still holds. Therefore, no co-occurrence relation is robust to filtering.

– The handover of work relation between individual \(a\) and \(b\) \((H(a, b))\) exists if there are two subsequent activities where the first is completed by \(a\) and the second by \(b\). This is also an example of non-robust intermediate structure for
filtering. Imagine we have $H(a, b)$ and $H(b, c)$. When $b$ is filtered, $a$ directly handed over to $c$, hence $H(a, c)$ must be deduced. This indicates the whole relations need to be recalculated.

The Integer Linear Programming (ILP) Miner uses language-based theory of regions in its discovery. The regions are produced from a prefix-closed language which is considered as the intermediate structure. As an example, we have $\log L = \{(a, b, c), (a, d, e)\}$. The prefix-closed language of $L$ is $\mathcal{L} = \{e, (a), (a, b), (a, d), (a, b, c), (a, d, e)\}$. It is clear that $\mathcal{L}$ is bigger than $L$. The prefix-closed language in region theory is one of the intermediate structures whose size is bigger than the log size.

While many intermediate structures can be identified when studying process mining techniques, we currently focus on the Directly Follows Relation (DFR). DFR is used in many process mining algorithms, including the most widely used process discovery techniques, i.e. Inductive Miner [8]. In the following we discuss how DB-XES is extended by a DFR table.

4.1 The DFR Intermediate Structure in DB-XES

Directly Follows Relation (DFR) contains information about the frequency with which one event class directly follows another event class in the context of a trace. Following the definition in [15], DFR is defined as follows.

**Definition 1 (Event Log).** Let $E$ be a set of events. An event log $L \subseteq E^*$ is a set of event sequences (called traces) such that each event appears precisely once in precisely one trace.

**Definition 2 (Event Attributes and Classifiers).** Let $E$ be a set of events and let $A$ be a set of attribute names.

- For any event $e \in E$ and name $a \in A$: $\#_a(e)$ is the value of attribute $a$ for event $e$. $\#_a(e) = \bot$ if there is no value.
Any subset $C \subseteq \{a_1, a_2, \ldots, a_n\} \subseteq A$ is a classifier, i.e., an ordered set of attributes. We define: $\#_c(e) = (\#_{a_1}(e), \#_{a_2}(e), \ldots, \#_{a_n}(e))$.

In the context of an event log there is a default classifier $DC \subseteq A$ for which we define the shorthand of event class $e = \#_{DC}(e)$.

**Definition 3 (Directly Follows Relation (DFR)).** Let $L \subseteq E^*$ be an event log. $x$ is directly followed by $y$, denoted $x > y$, if and only if there is a trace $\sigma = \langle e_1, e_2, \ldots, e_n \rangle \in L$ and $1 \leq i < n$ such that $e_i = x$ and $e_{i+1} = y$.

Translated to DB-XES, table $dfr$ consists of three important columns next to the id of the table, namely $eventclass_1$ which indicates the first event class in directly follows relation, $eventclass_2$ for the second event class, and $freq$ which indicates how often an event class is directly followed by another event class. Figure 3 shows the position of table $dfr$ in DB-XES. As DFR is defined on the event classes based on a classifier, every instance in table $dfr$ is linked to an instance of table $classifier$ in the log.

**Definition 4 (Table dfr).** Let $L \subseteq E^*$ be an event log, $X = \{e \mid e \in E\}$ is the set of event classes. $dfr \in X \times X \rightarrow \mathbb{N}$ where:

- $\text{dom}(dfr) = \{(x, y) \in X \times X \mid x > y\}$
- $dfr(x, y) = \sum_{\langle e_1, \ldots, e_n \rangle \in L} |\{i \in \{1, \ldots, n-1\} \mid e_i = x \land e_{i+1} = y\}|$

As mentioned before, the design choice to incorporate DFR as the intermediate structure is due to fact that DFR is used in the state-of-the-art process discovery algorithm. However, DB-XES can be extended into other intermediate structures such as eventually follows relations and no co-occurrence relations.

## 5 DFR Pre-Computation in DB-XES

Typically, process mining algorithms build an intermediate structure in memory while going through the event log in a single pass (as depicted in Figure 1(a)). However, this approach will not be feasible to handle huge event log whose size exceeds the computer memory. Moving the location of the event data from a file to a database as depicted in Figure 1(b) increases the scalability of process mining as the computer memory no longer needs to contain the event data. However, the ping-pong communication between the database and process mining tools is time consuming. Therefore, in this section, we show how DFR is pre-computed in DB-XES (Figure 1(c)). Particularly, we show how common processing tasks can be moved both in time and location, i.e. we show how to store intermediate structures in DB-XES and we show how these structures can be updated while inserting the data rather than when doing the process mining task. This paper focuses on a particular intermediate structure, namely the DFR, but the work trivially extends to other intermediate structures, as long as they can be kept up-to-date during insertion of event data in the database.

As mentioned in Section 4, the table $dfr$ in Figure 3 is the table in DB-XES which stores DFR values, furthermore, the table $log\_has\_dfr$ stores the context in
which the DFR exists, i.e. it links the DFR values to a specific log and classifier combination. The \textit{dfr} table is responsive to update operations, particularly when users insert new events to the log. In the following we discuss how the \textit{dfr} table is created and updated in DB-XES.

5.1 Creating Table \textit{dfr} in DB-XES

Suppose that there exists two entries in the \textit{trace\_has\_event} table with trace id \(\sigma\), event id’s \(e_i\) and \(e_{i+1}\) and sequence’s \(i\) and \(i+1\). The first event \(e_i\) is linked to an attribute \(\alpha\) with value \(a\) and the second event is linked to an attribute \(\alpha\) with value \(b\) while the log has a classifier based on attribute \(\alpha\). In DB-XES, we store the frequency of each pair \(a > b\) in the database rather than letting the discovery algorithm build it on-demand and in-memory. In other words, the directly follows relation is precomputed and the values can be retrieved directly by a process mining algorithm when needed.

To create table \textit{dfr}, we run three SQL queries. The first query is to obtain pairs of directly follows relations. For instance, if an event class \(a\) is directly followed by an event class \(b\) and this happens 100 times in the log, then there will be a row in table \textit{dfr} with value \((\text{df}_{r_1}, a, b, 100)\), assuming the id is \(\text{df}_{r_1}\). Furthermore, the second and third queries are to extract start and end event classes. We create an artificial start \((\top)\) and end \((\bot)\) event for each process instance. For example, if there are 200 cases where \(a\) happens as the start event class, there will be a row in \textit{dfr} with values \((\text{df}_{r_1}, \top, a, 200)\). Similarly, if \(b\) is the end event class for 150 cases, there will be a row in \textit{dfr} with values \((\text{df}_{r_1}, \bot, b, 150)\).

Technically, the SQL query contains big joins between tables \textit{trace\_has\_event}, \textit{event}, \textit{attribute}, \textit{log\_has\_trace}, \textit{log\_has\_classifier}, and \textit{classifier}. Such joins are needed to get pairs of event classes whose events belong to the same trace in the same log which has some classifiers. The SQL query mentioned below is a simplified query to obtain pairs of directly follows relations. To improve understandability, we use placeholders (\(< ... >\)) to abstract some details. Basically they are trivial join conditions or selection conditions to interesting columns.

```sql
1 SELECT id, eventClass1, eventClass2, count(*) as freq
2 FROM (
3     SELECT <...>
4     FROM (  
5         SELECT <...>
6         FROM trace\_has\_event as t1
7         INNER JOIN trace\_has\_event as t2
8         ON t1.trace\_id = t2.trace\_id
9         /* Here is to get consecutive events */
10         WHERE t1.sequence = t2.sequence - 1
11     ) as temptable,
12     attribute as a1, attribute as a2,
13     event as event1, event as event2,
```
We start with a self join in table `trace_has_event` (line 6-8) to get pairs of two events which belong to the same trace. Then we filter to pairs whose events happen consecutively, i.e. the sequence of an event is preceded by the other (line 10). The next step is obtaining the attribute values of these events. The attribute values are grouped based on the classifier in the log (line 16-17). This grouping is essential if the classifier is built from a combination of several attributes, for example a classifier based on the activity name and lifecycle. After grouping, we get a multiset of pairs of event classes. Finally, the same pairs are grouped and counted to have the frequency of how often they appeared in the log (line 1, 19).

5.2 Updating Table `dfr` in DB-XES

Rows in table `dfr` are automatically updated whenever users insert a new event through a trigger operation on table `trace_has_event` which is aware of an insert command. Here we consider two scenarios: (1) a newly inserted event belongs to a new trace in a log for which a `dfr` table exists and (2) a newly inserted event belongs to an existing trace in such a log. We assume such insertion is well-ordered, i.e. an event is not inserted at an arbitrary position.

Suppose that we have a very small log $L = [(a, b)]$, where we assume $a$ and $b$ refer to the event class of the two events in $L$ determined by a classifier $cl$ for which an entry $(L, cl, dfr_1)$ exists in the `log_has_dfr` table. This log only contains one trace (say $\sigma_1$) with two events that correspond to two event classes, namely $a$ and $b$. If we add to $L$ a new event with a new event class $c$ to a new trace different from $\sigma_1$ then such an event is considered as in the first scenario. However, if we add $c$ to $\sigma_1$ then it is considered as the second scenario.

In the first scenario, we update the start and end frequency of the inserted event type. In our example above, the rows in table `dfr` containing $(dfr_1, \top, c, f)$ and $(dfr_1, c, \bot, f)$ will be updated as $(dfr_1, \top, c, f + 1)$ and $(dfr_1, c, \bot, f + 1)$ with $f$ is the frequency value. If there is no such rows, $(dfr_1, \top, c, 1)$ and $(dfr_1, c, \bot, 1)$ will be inserted.

In the second scenario, we update the end frequency of the last event class before the newly inserted event class, and add the frequency of the pair of those two. Referring to our example, row $(dfr_1, b, \bot, f)$ is updated to $(dfr_1, b, \bot, f - 1)$. If there exists row $(dfr_1, c, \bot, f)$, it is updated to $(dfr_1, c, \bot, f + 1)$, otherwise $(dfr_1, c, \bot, 1)$ is inserted. Furthermore, if $(dfr_1, b, c, f)$ exists in table `dfr`, it is updated as $(dfr_1, b, c, f + 1)$, otherwise $(dfr_1, b, c, 1)$ is inserted.

By storing the intermediate structure in the database and updating this structure when events are inserted, we move a significant amount of computation time to the database rather than to the process analysis tool. This allows for
faster analysis with virtually no limits on the size of the event log as we show in
the next section.

6 Experiments

In this section we show the influence of moving both the event data and the
directly follows table to the database on the memory use and time consumption
of Inductive Miner \cite{8}. Next to the traditional in-memory processing of event
logs (Figure 1(a)), we consider two scenarios in DB-XES: (1) \textit{DB-XES without
DFR} where the intermediate result is computed during the discovery (Figure
1(b)) and (2) \textit{DB-XES with DFR} where the intermediate result is pre-computed
in the database (Figure 1(c)). We show that the latter provide scalability with
respect to data size and even improves time spent on actual analysis.

As the basis for the experiments, we use an event log from a real company
which contains 29,640 traces, 2,453,386 events, 54 different event classes and
17,262,635 attributes. Then we extend this log in two dimensions, i.e. we in-
crease (1) the number of event classes and (2) the number of traces, events and
attributes. We extend the log by inserting copies of the original event log data
with some modifications in the identifier, task name, and timestamp. In both
cases, we keep the other dimension fixed in order to get a clear picture of the
influence of each dimension separately on both memory use and CPU time. This
experiment was executed on the machine with processor Intel(R) Core(TM) i7-
4700MQ and 16GB of RAM.

6.1 Memory Use

In Figure 4(a), we show the influence of increasing the number of event classes
on the memory use of the Inductive Miner. The Inductive Miner makes a linear
pass over the event log in order to build an object storing the direct succession
relation in memory. In theory, the direct succession relation is quadratic in the
number of event classes, but as only actual pairs of event classes with more than
one occurrence are stored and the relation is sparse, the memory consumption
scales linearly in the number of event classes as shown by the trendlines. It is
clear that the memory use of DB-XES is consistently lower than XES. This is
easily explained as there is no need to store the event log in memory. The fact
that DB-XES with DFR uses more memory than DB-XES without DFR is due
to the memory overhead of querying the database for the entire DFR table at
once.

In Figure 4(b), we present the influence of increasing the number of events,
traces and attributes while keeping the number of event classes constant. In this
case, normal XES quickly uses more memory than the machine has while both
DB-XES implementations show no increase in memory use with growing data
and the overall memory use is less than 50 MB. This is expected as the memory
consumption of the Inductive Miner varies with the number of event classes only,
i.e. the higher frequency values in the \textit{dfr} table do not influence the memory use.
6.2 CPU Time

We also investigated the influence of accessing the database to the CPU time needed by the analysis, i.e. we measure the time spent to run the Inductive Miner. In Figure 5(a), we show the influence of the number of event classes on the CPU time. When switching from XES files to DB-XES without DFR, the time...
needed to do the analysis increases considerably. This is easily explained by the overhead introduced in Java by initiating the query every time to access an event. However, when using DB-XES with DFR, the time needed by the Inductive Miner decreases, i.e. it is faster to obtain the \textit{dfr} table from the database than to compute it in memory.

This effect is even greater when we increase the number of traces, events and attributes rather than the number of event classes as shown in Figure 5(b). DB-XES with DFR shows a constant CPU time use, while normal XES shows a steep linear increase in time use before running out of memory. DB-XES without DFR also requires linear time, but is several orders of magnitude slower (DB-XES without DFR is drawn against the right-hand side axis).

In this section, we have proven that the use of relational databases in process mining, i.e DB-XES, provide scalability in terms of memory use. However, accessing DB-XES directly by retrieving event data elements on demand and computing intermediate structures in ProM is expensive in terms of processing time. Therefore, we presented DB-XES with DFR where we moved the computation of the intermediate structure to the database. This solution provides scalability in both memory and time.

We have implemented this solution as a ProM plug-in which connects DB-XES and Inductive Miner algorithm. We name the plug-in as \textit{Database Inductive Miner} and it is distributed within the DatabaseInductiveMiner package (https://svn.win.tue.nl/repos/prom/Packages/DatabaseInductiveMiner/Trunk/).

7 Conclusion and Future Work

This paper focuses on the issue of scalability in terms of both memory use and CPU use in process discovery. We introduce a relational database schema called DB-XES to store event data and we show how directly follows relation can be stored in the same database and be kept up-to-date when inserting new events into the database.

Using experiments on real-life data we show that storing event data in DB-XES not only leads to a significant reduction in memory use of the process mining tool, but can even speed up the analysis if the pre-processing is done in the right way in the database upon insertion of the event data.

For the experiments we used the Inductive Miner, which is a state-of-the-art process discovery technique. However, the work trivially extends to other process discovery techniques, as long as we can identify an intermediate structure used by the technique which can be updated when inserting new events into the database.

The work presented in this paper is implemented in ProM. The plug-in paves a way to access pre-computed DFR stored in DB-XES. These DFR values are then retrieved and processed by Inductive Miner algorithm.

For future work, we plan to implement also the event removal and intermediate structures which robust to filtering. The intermediate structures will be
kept live under both insertion and deletion of events where possible. Furthermore, we aim to further improve the performance through query optimization and indexing.

References


On Marrying Model-Driven Engineering and Process Mining: A Case Study in Execution-based Model Profiling

Alexandra Mazak and Manuel Wimmer

Business Informatics Group (BIG),
Institute of Software Technology and Interactive Systems,
TU Wien, Favoritenstrasse 9-11, 1040 Vienna, Austria
{mazak,wimmer}@big.tuwien.ac.at
http://www.big.tuwien.ac.at

Abstract. In model-driven engineering (MDE), models are mostly used in prescriptive ways for system engineering. While prescriptive models are indeed an important ingredient to realize a system, for later phases in the systems’ lifecycles additional model types are beneficial to use. Unfortunately, current MDE approaches mostly neglect the information upstream in terms of descriptive models from operations to design, which would be highly needed to improve systems continuously. To tackle this limitation, we propose execution-based model profiling as a continuous process to improve prescriptive models at design-time through runtime information by incorporating knowledge in form of profiled metadata from event logs generated during the execution of a code model. For this purpose we combine techniques of process mining (PM) with runtime models of MDE. In the course of a case study, we implement a preliminary prototype of our framework based on a traffic light system example which shows the feasibility and benefits of our execution-based model profiling approach.

1 Introduction

In model-driven engineering (MDE), models are put in the center and used throughout the software development process, finally leading to an automated generation of the software systems [13]. In the current state-of-practice in MDE [2], models are used as an abstraction and generalization of a system to be developed. By definition a model never describes reality in its entirety, rather it describes a scope of reality for a certain purpose in a given context. Thus, models are used as prescriptive models for creating a software system [18]. Such models@design.time determine the scope and details of a domain of interest to be studied. Thereby, different aspects of the domain or of its solution can be taken into account. For this purpose different types of modeling languages (e.g., state charts, class diagrams, etc.) may be used. It has to be emphasized that engineers typically have the desirable behavior in mind when creating a system, since they are not aware in these early phases of the many deviations that may take place at runtime [3].

According to Brambilla et al. [2] the implementation phase deals with the mapping of prescriptive models to some executable systems and consists of three levels: (i) the
modeling level where the models are defined, (ii) the realization level where the solutions are implemented through artifacts that are used in the running system, and (iii) the automation level where mappings from the modeling to the realization phase are made. Thus, the flow is from models down to the running realization through model transformations.

While prescriptive or design models are indeed a very important ingredient to realize a system, for later phases in the system’s lifecycle additional model types are needed. Therefore, descriptive models may be employed to better understand how the system is actually realized and how it is operating in a certain environment. Compared to prescriptive models, these other mentioned types of models are only marginal explored in the field of MDE, and if used at all, they are built manually. Unfortunately, MDE approaches have mostly neglected the possibility to describe an existing and operating system which may act as feedback for design models. As theoretically outlined in [12], we propose model profiling as a continuous process (i) to improve the quality of design models through runtime information by incorporating knowledge in form of profiled metadata from the system’s operation, (ii) to deal with the evolution of these models, and (iii) to better anticipate the unforeseen. However, our aim is not to “re-invent the wheel” when we try to close the loop between downstream information derived from prescriptive models and upstream information in terms of descriptive models. There exist already promising techniques to focus on runtime phenomena, especially in the research field of Process Mining (PM) [3]. Thus, our model profiling approach in its first version follows the main idea of combining MDE and PM. The contribution of this paper is to present a unifying architecture for a combined but loosely-coupled usage of MDE approaches and PM techniques.

The remainder of this paper is structured as follows. In the next section, we present a unified conceptual architecture for combining MDE with PM frameworks. In Section 3, we present a case study in execution-based model profiling based on a traffic light system example. In Section 4, we present recent work related to our approach and discuss its differences. Finally, we conclude this paper by an outlook on our next steps in Section 5.

2 Marrying Model-Driven Engineering and Process Mining

In this section, we briefly describe the main building blocks of both, MDE as well as PM, necessary for the context of this paper, before we present a unifying architecture for their combined but loosely-coupled usage.

2.1 Prerequisites

Model-driven Engineering (MDE). In each phase of an MDE-based development process “models” (e.g., analysis models, design models) are (semi-)automatically generated by model-to-model transformations (M2M) that take as input those models that were obtained in one of the previous phases. In the last step of this process the final code is generated using model-to-text transformation (M2T) from the design model [2]. These transformation engineering aspects are based on the metamodels of a modeling
language, which provide the abstract syntax of that language. This syntax guarantees
that models follow a clearly defined structure, and it forms the basis for applying operations on models (e.g., storing, querying, transforming, checking, etc.).

As described in [2], the semantics of a modeling language can be formalized by (i) giving denotational semantics by defining a mapping from the modeling language to a formal language, (ii) giving operational semantics by defining a model simulator (i.e., for implementing a model execution engine), or (iii) giving translational semantics by defining, e.g., a code generator for producing executable code. In order to generate a running system from models, they must be executable. This means, a model is executable when its operational semantics is fully specified [2]. However, executability depends more on the used execution engine than on the model itself. The main goal of MDE is to get running systems out of models.

In our approach, we consider executable modeling languages which explicitly state “what” the runtime state of a model is, as well as all possible events that can occur during execution [1]. These executable modeling languages not only provide operational semantics for interpreters, but also translational semantics in form of code generators to produce code for a concrete platform to realize the system.

**Process Mining (PM).** It combines techniques from data mining and model-driven Business Process Management (BPM) [3]. In PM, business processes are analyzed on the basis of event logs. Events are defined as process steps and event logs as sequential ordered events recorded by an information system [15]. This means that PM works on the basis of event data instead of designed models and the main challenge is to capture behavioral aspects. Thereby, specialized algorithms (e.g., the $\alpha$-algorithm) produce a Petri net, which can be easily converted into a descriptive model in form of a process model. To put it in a nutshell, there is a concrete, running system which is producing logs and there are algorithms used to compute derived information from those logs.

Generally in PM, event logs are analyzed from a process-oriented perspective using GPLs (e.g., UML, Petri nets) [4]. There are three main techniques in PM: (i) the well-established discovery technique by which a process model can be automatically extracted from log data [3], (ii) the conformance checking technique, which is used to connect an existing process model with an event log containing data related to activities (e.g., business activities) of this process [11], and (iii) the enhancement technique which is used to change or extend a process model by modifying it (i.e., repair), or by adding a new perspective to this model (i.e., extension) [3]. In recent work, van der Aalst already brings together PM with the domain of software engineering. For instance in [10], the authors present a novel reverse engineering technique to obtain real-life event logs from distributed software systems. Thereby, PM techniques are applied to obtain precise and formal models and to monitor and improve processes by performance analysis and conformance checking.

### 2.2 Unifying Conceptual Architecture

In this section, we combine MDE with PM by presenting a unifying conceptual architecture. The alignment of these two different research fields may help us, e.g., to verify
if the mapping feature of design models is really fulfilled, or if important information generated at runtime is actually missing in the design model.

Fig. 1 presents an overview of this architecture. On the left-hand side there is the prescriptive perspective, where we use models for creating a system, whereas on the right-hand side there is the descriptive perspective where models are extracted from running systems. In the following, we describe Fig. 1 from left to right. The starting point is the design language specification at the metamodeling level which defines the syntax as well as the semantics of the language. The design model at the modeling level describes a certain system and conforms to the design language. In our approach, this design model describes two different aspects of the problem or the solution: (i) the static aspect, which is used to describe the main ingredients of the modeled entity and their relationships, and (ii) the dynamic aspect, which describes the behavior of these ingredients in terms of events and interactions that may occur between them [2]. For the vertical transition from the model level to the realization level (i.e., the process of transforming models into source code), we use code generation at the automation level. Finally, at the realization level the running software relies on a specific platform for its execution.

At the right-hand side of Fig. 1 (at the top right), we present a logging metamodel—the so-called observation language. This metamodel defines the syntax and semantics of the (event) logs we want to observe from the running system. In particular, we derive this metamodel from the operational semantics of the design language. This means that this observation metamodel can be derived from any modeling language that can be equipped with operational semantics. Fig. 1 indicates this transformation at the metamodel level. The observation model, which conforms to the observation language, thumbs the logs at runtime and provides these logs as input for any kind of tools used for checking purposes of non-functional properties (e.g. performance, correctness, ap-
propriateness, etc.). This means that we can transform a language-specific observation model to a workflow file and import it, e.g., in a PM tool as presented in our case study.

3 Case Study: Execution-based Model Profiling

In this section, we perform an empirical explanatory case study based on the guidelines introduced in [14]. The main goal is to evaluate if current approaches for MDE and PM may be combined in a loosely-coupled way, i.e., both can stay as they are initially developed, but provide interfaces to each other to exchange the necessary information to perform automated tasks. In particular, we report on our results concerning a fully model-driven engineered traffic light system which is enhanced with execution-based model profiling capabilities. All artifacts of the case study can be found on our project website.¹

3.1 Research questions

As mentioned above, we performed this study to evaluate the feasibility and benefits of combining MDE and PM approaches. More specifically, we aimed to answer the following research questions (RQ):

1. **RQ1—Transformability**: Is the operational semantics of the modeling language rich enough to automatically derive observation metamodels?
2. **RQ2—Interoperability**: Are the observation metamodels amenable for PM tools by reusing existing process mining formats?
3. **RQ3—Usefulness**: Are the generated model profiles resulting from the observation model sufficient enough for runtime verification?

3.2 Case Study Design

Requirements. As an appropriate input to this case study, we require a system which is generated by an MDE approach equipped with an executable modeling language, i.e., its syntax and operational semantics are clearly defined and accessible. Furthermore, the approach has to provide translational semantics based on a code generator which may be extended by additional concerns such as logging. Finally, the execution platform hosting the generated code must provide some means to deal with execution logs.

Setup. To fulfill the stated requirements, we selected an existing MDE project concerning the automation controller of a traffic light system. This system consists of several components, e.g., lights (green, yellow, red) for cars and pedestrians, as well as a controller of the system. We modelled this example by using a UML-based design language named **Class/State Charts** (CSC) resulting in a class diagram and a timed state machine as prescriptive models (cf. Fig. 2). While the state machine shows all possible and valid transitions/states within this example, the class **TrafficLightController** specifies the blinkcounter (bc) and the different lights which can be on or off. The models of this example have been developed by using the Embedded Engineer.² We use the

¹ [http://www.sysml4industry.org/?page_id=722](http://www.sysml4industry.org/?page_id=722)
² [http://www.lieberlieber.com](http://www.lieberlieber.com)
Embedded Engineer also to produce Python code from the traffic light model. The code can be executed on a single-board computer. We use as specific execution platform a Raspberry Pi (see Fig. 3, at the bottom left). It has to be noted that we aimed for full code generation by exploiting a model library which allows to directly delegate to the GPIO module (i.e., input/output module) of the Raspberry Pi.

3.3 Results

In this subsection, we present the results of applying the approach presented in Section 2.2 for the given case study setup. First, we show the general architecture to realize execution-based model profiling for the traffic light system example. Subsequently, several details of the realization are presented by focussing on the observation metamodel as well as the usage of an established PM tool.

Technical Realization at a Glance. Our prototypical realization of execution-based model profiling considers the execution logs of a running code model as the experimental frame. Fig. 3 gives an overview of its implementation. We extend the code generator to produce Python code which enables to report logs to a log recording service implemented as MicroService provided by an observation model repository. For data exchange between the running system and the log recording service, we use JSON, which means that the JSON data transferred to the MicroService is parsed into log entry elements in the repository. We use Neo4EMF\(^3\) to store the execution logs in a NoSQL database for further analysis. In order to be able to use established PM tools, we generate XML files from the recorded logs (i.e., the observation model). For first evaluation purposes, we import this files in PromLite.\(^4\) The use of this PM tool enables us to generate different Petri net (PN) models from the recorded logs. In more detail, we use ATL [20] as transformation tool to transform an observation model to a workflow instance model and import it in ProM to run the PN discoverer.

The Observation Metamodel. According to PM techniques, we consider an observation model as an event log with a start and end time registered as a sequences of transactions that having already taken place. However, we do not receive event logs

\(^3\)http://www.neoemf.com

\(^4\)http://www.promtools.org/doku.php?id=promlite
from an executed process model (i.e., the activities of a business process in an ordered manner), rather we receive the traces from transformed log messages of an executed code model.

Fig. 4 shows the observation metamodel derived from the operational semantics of the excerpt of UML which is considered in the context of this case study. It illustrates that changes at runtime (rt) are basically value updates for attributes of the UML class diagram as well as updates concerning the current active state of the UML state machine (cf. Fig. 4, these elements are marked with the \( \text{rt} \) stereotype). The class Log represents a logging session of a certain running software system with a registered observationStart and an observationEnd. The class Log consists of log entries with a unique id and a timeStamp for ordering purpose (i.e., showing when the entry was recorded). The LogEntry either registers an AttValueChange or a CurStateChange. In case of a CurStateChange the LogEntry considers the predecessor (pre) and successor (succ) states. If an attribute has changed the LogEntry includes the additional information about the preValue and postValue.

**Generated Model Profiles.** For generating different model profiles from the observation model, we employ existing PM tools. For this purpose, we have reverse engineered the XML Schema of the workflow log language\(^5\) into a metamodel, which allows to translate our language-specific observation model into workflow instances by defining different model transformations based on the information which should be discovered. For our case study, we have first implemented a model transformation in ATL which considers the state occurrences of system runs, i.e., it is a view on the observation model just considering the CurStateChange elements. By this, we can check if the state machine is realized as intended by the code generator and if it executes on the realization level as specified. In particular, their equivalence can be check semantically by comparing the state space of the state machine with the state space of the observed Petri net or also syntactically by defining bi-directional transformation rules which can be used to check the consistency of two heterogeneous models [23]. Second, we have

\(^5\)http://www.processmining.org/WorkflowLog.xsd
developed another ATL transformation which extracts for each attribute a workflow instance which contains the sequence of AttValueChange instances. By this, we can extract the shape of the values stored in the attribute and enrich the model with this kind of information, or check if certain value constraints are fulfilled during execution. For instance, for the blink counter \( (bc) \) attribute we have derived a PN which explicitly shows a loop counting from zero to six. All the generated artefacts can be found on our project website.

### 3.4 Interpretation of Results

**Answering RQ1.** The operational semantics could be transferred into an observational viewpoint. By generating a change class for every element in the design metamodel which is annotated with the \( \langle rt \rangle \) stereotype, we are able to provide a language to represent observations of the system execution. This language can be also employed to instrument the code generator in order to produce the necessary logging statements as well as to parse the logs into observation model elements.

**Answering RQ2.** By developing transformations from the language-specific observation metamodels to the general workflow-oriented formats of existing PM tools, we could reuse existing PM analysis methods for MDE approaches in a flexible manner. Not only the state/transition system resulting from the state machine can be checked between implementation and design, but also other mining tasks could be achieved such as computing value shapes for the given attributes of the class diagram. Thus, we conclude that it is possible to reuse existing formats for translating the observations, however, different transformations may be preferred based on the given scenario.
Answering RQ3. For runtime verification, we took as input transformed event logs (i.e., selected state changes as a workflow file) and employed the \( \alpha^+\) algorithm of PromLite 1.1 to derive a PN. This generated PN exactly corresponds to the state machine, as shown in Fig. 2 on the right hand side. We are therefore convinced that the state machine is realized as intended by the code generator. Similarly, we have done this for variable changes. As output we extracted a value shape \([0..6]\) stored in the attribute blink counter. By doing so we demonstrated, that we are also able to enrich the initial class diagram with runtime information in terms of profiled metadata. Finally, we manually implemented random error-handling states in the Python code model (not in the design model) to show that these errors are one-to-one reflected in the generated PN. By applying bi-directional transformations, these additional states may be also propagated to the state machine model in order to complete the specification for error-handling states which are often neglected in design models [5].

3.5 Threats to Validity

To critically reflect our results, we discuss several threats to validity of our study. First, in the current realization of our approach we do not consider the instrumentation overhead which may increase the execution time of the instrumented application. Of course, this may be critical for timed systems and has to be validated further in the future. Second, the current system is running as a single thread which means we are not dealing with concurrency. Extensions for supporting concurrency may result in transforming the strict sequences in partially ordered ones. Third, we assume to have a platform which has network access to send the logs to the micro service. This requirement may be critical in restricted environments and measurements of network traffic have to be done. Finally, concerning the generalizability of the results, we have to emphasize that we currently only investigated one modeling language and one execution platform. Therefore, more experiments are needed to verify if the results can be reproduced for a variety of modeling languages and execution platforms.

4 Related Work

We consider model profiling as a very promising field in MDE and as the natural continuation and unification of different already existing or emerging techniques, e.g., data profiling [7], process mining [3], complex event processing [6], specification mining [5], finite state automata learning [8], as well as knowledge discovery and data mining [9]. All these techniques aim at better understanding the concrete data and events used in or by a system and by focusing on particular aspects of it. For instance, data profiling and mining consider the information stored in databases, while process mining, FSA learning and specification mining focus on chronologically ordered events. Not to forget models@run.time, where runtime information is propagated back to engineering. There are several approaches for runtime monitoring. Blair et al. [22] show the importance of supporting runtime adaptations to extend the use of MDE. The authors propose models that provide abstractions of systems during runtime. Hartmann et al. [21] go one step further. The authors combine the ideas of runtime models with
reactive programming and peer-to-peer distribution. They define runtime models as a stream of model chunks, like it is common in reactive programming.

Currently, there is emerging research work focusing on runtime phenomena, runtime monitoring as well as discussing the differences between descriptive and prescriptive models. For instance, Das et al. [16] combine the use of MDE, run-time monitoring, and animation for the development and analysis of components in real-time embedded systems. The authors envision a unified infrastructure to address specific challenges of real-time embedded systems’ design and development. Thereby, they focus on integrated debugging, monitoring, verification, and continuous development activities. Their approach is highly customizable through a context configuration model for supporting these different tasks. Szvetits and Zdun [17] discuss the question if information provided by models can also improve the analysis capabilities of human users. In this context, they conduct a controlled experiment. Heldal et al. [18] report lessons learned from collaborations with three large companies. The authors conclude that it is important to distinguish between descriptive models (used for documentation) and prescriptive models (used for development) to better understand the adoption of modeling in industry. Last but not least, Kühne [19] highlights the differences between explanatory and constructive modeling, which give rise to two almost disjoint modeling universes, each of it based on different, mutually incompatible assumptions, concepts, techniques, and tools.

5 Conclusion and Next Steps

In this paper, we pointed to the gap between design time and runtime in the current MDE approaches. We stressed that there are already well-established techniques considering runtime aspects in the area of PM and that it is beneficial to combine these approaches. Therefore, we presented a unifying conceptual architecture for execution-based model profiling, where we combined MDE and PM. We built this approach upon traditional activities of MDE such as design modeling, code generation, and execution and demonstrated it for the traffic light system case study. While the first results seem promising, there are still several open challenges, which we discussed in the threats to validity subsection of the case study. As next steps we will focus on reproducing our current results with other case studies, e.g., by considering a domain-specific modeling language for an elevator system [1].

Acknowledgment

This work has been supported by the Austrian Science Fund (FWF): P 28519-N31 and by the Austrian Research Promotion Agency (FFG) within the project “InteGra 4.0 - Horizontal and Vertical Interface Integration 4.0”.

References

Improving Process Model Precision by Loop Unrolling

David Sánchez-Charles¹, Marc Solé¹, Josep Carmona², Victor Muntés-Mulero¹

1 CA Strategic Research Labs, CA Technologies, Spain
   David.Sanchez,Marc.SoleSimo,Victor.Muntes@ca.com
2 Universitat Politècnica de Catalunya, Spain
   jcarmona@cs.upc.edu

Abstract. Despite the advent of scalable process mining techniques that can handle both noisy and incomplete real-life event logs, there is a lack of scalable algorithms capable of handling a common cause of model underfitting: when the same activity in the log in fact behaves differently depending on the number of occurrences in a particular trace. This paper proposes a simple scalable technique to identify these cases and successfully mine better process models from event logs. The technique has been implemented and evaluated on well-known benchmarks in the literature.

1 Introduction

Process discovery techniques strive to derive models that are expected to be good under four quality dimensions: fitness, precision, generalization and simplicity [4]. Hence, these are multi-objective techniques that search in a large solution space, where typically not one but many optimal solutions exist. In practice, each discovery technique puts the emphasis in a proper subset of dimensions; for instance, techniques based in the theory of regions focus on deriving fitting and precise models, while simplicity and generalization is not always guaranteed. Another example is the recent block-based techniques that recently appeared [6, 7], where structured, fitting, generalized and simple process models are preferred.

The techniques from [6, 7] are the driving force of this work. On the one hand, they are among the few scalable process discovery technique that can derive structured process models. This has made [6, 7] one of the most popular techniques for process discovery nowadays. However, as mentioned in [3], these techniques can sacrifice precision significantly for the sake of deriving a fitting structured model (see the example in Section 1.1). The alternative offered in [3] is to use evolutionary techniques, which are far from scalable. Instead, the technique proposed in this paper represent a fresh look at this problem, amending (when possible) process models derived from the technique in [6, 7] as a simple post-processing step, based on unrolling loops in the model whenever the number of loop iterations found in the event log satisfy certain criteria. Next section illustrates the intuition behind the technique of this paper.
1.1 Label splitting as loop unrolling to improve precision

Consider the model Figure 1.a, which was discovered by considering the trace $\sigma = ABCADCBACDABCADCBACADE$. It is hard to notice that the precision of this model could be improved: Activities $A$, $B$ and $D$ can be found in any ordering and hence the parallel construct is appropriate, and trace $\sigma$ hints that the iterative approach might be a good candidate for describing such a process. Nevertheless, a further analysis shows that there is still place for improvement.

In this paper, we propose to unroll iterative parts of a process to check if there are hidden relations between the activities that are hindered by the limitation of only having one single copy of the activity in the model. See Figure 1.b for an example of such unrolling. In this particular case, we have chosen to repeat the iterative structure so we are forcing to execute its subprocess twice in each iteration. A replay of trace $\sigma$ on this new process model highlights that activities $B$ and $D$ were never mutually exclusive. And hence, one could discover that the process model of Figure 1.c might be more precise in describing $\sigma$.

1.2 Related work

Different approaches exist in the literature for the problem of label splitting in the context of process mining. We will focus here in recent approaches, and will illustrate the different nature of the technique of this paper with respect to
them. The heuristic techniques in [10, 8] rely on a window local search approach to define the duplication of certain candidate activities. This process is done in the model itself ([10]) or as a refinement of the input log ([8]). By focusing on the loops of a process model, the technique of this paper complements these approaches.

Alternatively, global approaches can be found in [9, 5]. These global methods rely on the use of unfolding of the process model ([9]) and a later optimization technique to fold back activities, or search for special states of the underlying state space of the model ([5]), followed by a clustering strategy to merge them heuristically. By relying on complex representations and techniques (unfoldings or state spaces can be exponential on the size of the models), these approaches cannot be applicable for large inputs.

## 2 Definitions and notation

### Definition 1

A **process model**, or simply **process**, $N = (A, C, E)$ is a directed graph consisting of activities $A$, control elements $C$ and edges $E$. Edges connect activities and control elements. Control elements define the behavior of the process model, and are of any of the following types start, finish, split-choice, join-choice split-parallel, join-parallel. The only condition over the graph structure is that all maximal paths must start and end with a start and a finish control flows. A **subprocess** of $N$ is any valid process model $(A', C', E')$, with $A \supseteq A'$, $C \supseteq C'$, in which $E'$ is defined as all edges in $E$ that connects any pair of elements in $A'$ and/or $C'$.

Processes are a graph representation of a potentially infinite set of sequences of events, which is denoted by $L(N)$. Generally, all paths from the initial start control element to the end traverse a sequence of activities. Such sequences are the elements of $L(N)$. Although the graphical notation used for representing processes is irrelevant in terms of the results presented in this paper, Business Process Modelling Notation will be used to improve understandability.

### Definition 2

**Structured process** imposes extra conditions on the control elements of a process: all split-parallel nodes (resp. split-choice) must have a unique corresponding join-parallel node (resp. join-choice) such that all paths connecting these two nodes must visit zero or two of any other pair of control elements. This correspondence is unique in the sense that if two split nodes $u$ and $v$ have the same corresponding join node, then $u$ and $v$ are the same node.

This definition allows us to consider structured processes as smaller subprocesses or individual activities that are interconnected via edges or control elements. Due to the soundness of structured processes, some notions can be easily described.

### Definition 3

An **iterative subprocess** or loop $l$ is the combination of two subprocesses that describe a process that can be repeated. The **forward path**
of \( l \) (\( \text{fwd}(l) \)) is the subprocess that must be executed at least once during the execution \( l \). Whereas the \textbf{backward path} of \( l \) (\( \text{back}(l) \)) is the subprocess such that its execution enforces the loop to re-execute \( \text{fwd}(l) \).

From now on we will consider all process models to be structured. Importantly, structured processes allow us to map particular events in the trace to a subprocess in the process model. Allowing us to define the following notions:

**Definition 4.** Given a process model \( N \) with a loop \( l \) and a trace \( \sigma \in \mathcal{L}(N) \), we define \( E_l(\sigma) \) as the number of times \( \text{fwd}(l) \) is executed during the execution of \( \sigma \).

**Definition 5.** Let \( l \) be a loop of a process model \( N \) and \( \sigma \) a trace accepted by \( N \). We define the \textbf{projection of \( \sigma \) to \( l \)} (denoted by \( \sigma|_l \)) as the result of keeping the events that are mapped into activities contained in \( l \) after a replay of \( \sigma \) in \( N \). Moreover, we define the \textbf{projection of \( \sigma \) to the exit condition of \( l \)} (denoted by \( \sigma|_{\text{Exit}(l)} \)) as keeping the events that are mapped into activities of \( N \) that cannot coexist with the execution of \( l \). In particular, all activities contained in \( l \) and any other concurrent activity are erased by this projection.

Considering again the process model of Figure 1.a and trace \( \sigma = ABCAD-CBACDABCADCBACADE \), we have a loop structure \( l \) consisting of: as the forward path, activities \( A \), \( B \) and \( D \) that can be executed concurrently but \( B \) and \( D \) are mutually exclusive; and activity \( C \) as its backward path. The forward path was executed \( E_l(\sigma) = 8 \) times; \( \sigma|_l = \sigma - \{E\} \) whereas \( \sigma|_{\text{Exit}(l)} = E \). Any execution of activity \( E \) clearly indicates that any event after event \( E \) is not part of the loop.

**Definition 6.** (Fitness and precision) Process mining techniques aim at extracting from a log \( L \) a process model \( N \) with the goal to elicit the real unknown process \( S \). By relating the behaviors of \( L \), \( \mathcal{L}(N) \) and \( S \), particular concepts can be defined [4]. A process model \( N \) fits \( L \) if \( L \subseteq \mathcal{L}(N) \). A process model is precise in describing a log \( L \) if \( \mathcal{L}(N) \setminus L \) is small.

Unless stated otherwise, we assume we deal with fitting process models. In case this condition is violated, we assume the process models are first aligned with current techniques to satisfy the fitness condition [1].

### 3 Label splitting with loop unrolling

Most discovery algorithms generate processes in which activities are not duplicated, forcing the algorithm to introduce loops when an activity is consistently occurring multiple times per trace. Unfortunately, this constraint may overgeneralize the resulting process model. Consider, for instance, trace \( \sigma = ABCA \). A technique like the ones in [6, 7] will produce an iterative process model even though the trace is not showing so clearly that behavior.

First we will describe the unrolling algorithm for improving the precision of loops that are not included in any other iterative process. The main idea of this
algorithm is to create a process model that forces the re-execution of the loop and then filters out unused elements. Finally, we will extend this algorithm for the case of nested loops.

3.1 Simple case: Unrolling of individual loops

The first process model of Figure 1.a depicts a process describing the log consisting of the trace ABCADCBACDABCADCBACADE. One may notice that, when replaying the log on the process model, the forward path (Activities A, B and D) is executed a multiple of two. Hence, we may force the process model to repeat the loop as in Figure 1.b. The unroll of a loop is precisely the process of making explicit this transformation.

Definition 7. A $k$-unroll of a loop $l$ is the process of substituting the loop for the subprocess defined by a loop structure with:

- A sequence of $k-1$ copies of the sequence $fwd(l);back(l)$ as the forward path of the new loop structure,
- finishing the aforementioned sequence with another copy of $fwd(l)$;
- The back($l$) is maintained as the backward path of the new loop structure.

In Figure 1.b, a 2-unroll of the iterative process is performed. In this case, the subprocess containing activities A, B and D is the forward path, whilst activity C is the backward path. And hence, its 2-unroll produces a loop structure with the sequence $AND(A OR(B, D)) C AND(A OR(B, D))$ as the forward path and maintains C as the backward path.

Proposition 1. Given a process model $N$ describing the log $L$, a $k$-unrolling ($k > 1$) of a loop $l$ increases the precision of the model.

Besides, if the process model fits log $L$ and $k$ is a divisor of the greatest common divisor (gcd) of the number of executions per trace of the forward path of $l$, then the $k$-unrolled process also fits log $L$.

Proof. Let $l$ be a loop of $N$ and let $N_k$ be a $k$-unroll of $l$ with $k > 1$. By construction of the $k$-unroll, we can ensure that any trace is an element of the language of $N$ such that the forward path of $l$ is executed a multiple of $k$ times. I.e.

$$\mathcal{L}(N_k) = \{ \sigma \in \mathcal{L}(N) \mid E_l(\sigma) \text{ is divisible by } k \}$$

We will show that $\mathcal{L}(N_k) \subseteq \mathcal{L}(N)$ and hence, based on Definition 6, $N_k$ improves the precision of process model $N$. Let $\sigma'$ be a trace accepted by the process $N$ that visits exactly once the forward path of the loop $l$, and hence the backward path of $l$ is never visited. Since 1 is not a multiple of $k$, we can ensure that $\sigma'$ is not an element of $\mathcal{L}(N_k)$ and hence $\mathcal{L}(N_k) \setminus L$ is a non-trivial subset of $\mathcal{L}(N) \setminus L$ and therefore the precision of $N'$ is bigger than the precision of $N$.

Besides, let $k$ be a divisor of the greatest common divisor of the number of executions per trace of the forward path of $l$, $N$ a process model that fits log $L$ and $N_k$ the $k$-unroll of the process model $N$. Let $\sigma'$ be a trace of the log $L$. Since
\( N \) fits \( \log L \), the trace \( \sigma' \) is in the language of the process model. Moreover, the number of executions of the forward path of \( l \) is a multiple of \( k \) and hence the trace \( \sigma' \) is also an element of the language of \( L \). Therefore, \( N_k \) fits \( \log L \). □

Once all loops have been unrolled, some activities and structures of the resulting process model may be redundant or unused and can be removed or simplified allowing for further improvement on the precision of the process model. The first process model of Figure 2 describes traces \( ACADBCBD, BCBDACBD \) and \( BCAD \). Such process may benefit from a 2-unroll as shown in the second process model. Besides, a replay of the three traces highlight that split choices between \( C \) and \( D \) are unnecessary: starting with \( C \), activities \( C \) and \( D \) alternate in the execution of the process model. The last process model of Figure 2 depicts the process model after pruning unused paths.

![Fig. 2. A process model describing the traces ACADBCBD, BCBDACBD and BCAD.](image)

### 3.2 General case: Unrolling of nested loops

Structured subprocesses allow process models to have nested loops structures. This poses a problem for deciding the number of unrolls, as the number of executions of the forward path per trace may be interleaved across embedded loops. The process model from Figure 3 depicts a process with a nested loop that accepts trace \( ABBBABB \). If we follow the count executions of the forward path, then activity \( B \) is recommended to be unrolled 6 times, even though it has never been executed 6 times in a row.
Instead of considering the number of executions per trace, we may count the number of consecutive executions of a forward path. In the particular case of trace $ABBBABBB$, the forward path $B$ is consecutively executed 3 times at two different points in the trace, whilst the forward path consisting of activities $A$ and $B$ is consecutively executed 2 times. Definition 8 formalises this concept by counting the number of executions on maximal subtraces contained in the loop subprocess.

**Definition 8.** Let $l$ be a loop structure of the process model $N$, and $\sigma$ a trace accepted by $N$. Then we define the set of continuous executions of the loop $l$ in the trace $\sigma$ as

$$CE_l(\sigma) = \left\{ n \mid \exists \sigma_1, \sigma', \sigma_2 \text{ s.t. } \sigma = \sigma_1 \sigma' \sigma_2, \sigma'|_l \in L(l), \sigma'|_{Exit(l)} = \emptyset, E_l(\sigma'|_l) = n, \sigma' \text{ is maximal} \right\}$$

Informally, the set $CE_l(\sigma)$ represents a set of numbers, each one denoting continuous executions of $l$ in $\sigma$.

The combination of no exit condition and maximality of subtrace $\sigma'$ in Definition 8 ensures that we are splitting the trace $\sigma$ on chunks such that a continuous execution of the loop $l$ is not separated in two different subtraces. Besides, non-consecutive executions of the loop cannot be included in the same group as this would have shown some activities incompatible with the execution of the loop. Notice that activities that are executed concurrently alongside the iterative subprocess $l$ might be included in the subtrace $\sigma'$, but they are removed during the projection to the iterative subprocess $l$ and they are not part of the exit condition.

Consider trace $\sigma = ABBBABBB$ and the smaller loop, or $B$-loop, of process model 3. The exit condition of the $B$-loop is activity $A$, since any execution of that particular activity clearly shows that the execution is happening outside the $B$-loop. Hence, we may split $\sigma$ in two instances of $\sigma' = BBB$. Notice that we cannot extend it since then we would include an exit condition, and $\sigma'$ is accepted by the $B$-loop. And therefore, $CE_{B\text{-}loop}(\sigma) = 3$.

Similarly to the non-nested case, the language accepted by an unrolled process model can be described as a refinement on the language accepted by the original process model as depicted in Proposition 2.

**Proposition 2.** Let $N$ be a process model, and $l$ a loop subprocess of $N$. Let $N_k$ be any $k$-unroll of $l$. Then

$$L(N_k) = \{ \sigma \in L(N) \mid \forall n \in CE_l(\sigma), n \text{ is divisible by } k \}$$

**Proof.** The definition of the $k$-unroll of loop $l$ ensures that any execution of the loop $l$ executed a multiply of $k$ times the forward path of $l$ and hence any maximal subtrace $\sigma' \subseteq \sigma$ such that $\sigma'|_l \in L(l), \sigma'|_{Exit(l)} = \emptyset$ must satisfy that $k$ divides $E_l(\sigma')$. 

---

95
On the other hand, let $\sigma$ be a trace in $L(N)$ such that all continuous executions $CE_l(\sigma)$ are divisible by $k$. Then, $\sigma$ is also an element of the language $L(N_k)$. Suppose not, then the trace $\sigma$ violates any behavioural relation between a set of activities or the iterative process must finish before repeating $k$ times the forward path. Both cases are not possible. The former violates the fact that $\sigma \in L(N)$ and the latter violates the fact that $CE_l(\sigma)$ only contains multiples of $k$.

Proposition 3 is a direct consequence of Proposition 2, due to the hard constraint that $k$ divides all $n \in CE(t,l)$.

**Proposition 3 (Generalization of Proposition 1).** Given a process model $N$ describing the log $L$, a $k$-unrolling ($k > 1$) of a loop $l$ increases the precision of the process model. Besides, if the process model fits log $L$ and $k$ is a divisor of $CL(l,t)$ for all $t \in L(N)$, then the $k$-unrolled process model also fits log $L$.

Revisiting the example of trace $ABBBABBB$ and the process model of Figure 3, which contains a nested loop, a replay of the trace contemplates that the big loop is executed 2 times and the smaller loop is executed 3 times on each execution. Hence, we could perform a 3-unroll on the latter and a 2-unroll on the former. Doing so, we discover the second process model of Figure 3, and a second replay highlights the possibility of removing the unnecessary loop structures as illustrated by Figure 3.

![Fig. 3. Three process models describing the trace ABBBABBB.](image)

**4 Evaluation**

To evaluate our duplicate technique, we reuse an existing dataset comprising 15 small logs [10] whose source processes are well-known and reproduce behavior commonly found in real-life scenarios. Besides, we also considered the BPI Challenge 2012 dataset. This real-life log contains events describing the application process for a personal loan, or overdraft, within a global financing organization.
From all these events, we have only selected the events starting with \( W \) as they show actions performed by workers of the organization, instead of states in a fixed sequential process.

<table>
<thead>
<tr>
<th>Model</th>
<th>( P )</th>
<th>( T )</th>
<th>( \tau )</th>
<th>Precision</th>
<th>Fitness</th>
<th>( P )</th>
<th>( T )</th>
<th>( \tau )</th>
<th>Precision</th>
<th>Fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>alpha</td>
<td>14</td>
<td>21</td>
<td>6</td>
<td>0.6750</td>
<td>0.7386</td>
<td>12</td>
<td>19</td>
<td>3</td>
<td>0.70</td>
<td></td>
</tr>
<tr>
<td>betaSimpl</td>
<td>9</td>
<td>15</td>
<td>3</td>
<td>0.8333</td>
<td>0.9130</td>
<td>10</td>
<td>8</td>
<td>2</td>
<td>1.00</td>
<td></td>
</tr>
<tr>
<td>Fig5p1AND</td>
<td>5</td>
<td>6</td>
<td>1</td>
<td>0.7000</td>
<td>1.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fig5p1OR</td>
<td>9</td>
<td>14</td>
<td>6</td>
<td>0.6746</td>
<td>0.85</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fig6p19</td>
<td>10</td>
<td>15</td>
<td>8</td>
<td>0.6667</td>
<td>0.83</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fig6p10</td>
<td>15</td>
<td>24</td>
<td>13</td>
<td>0.6282</td>
<td>0.76</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fig6p25</td>
<td>22</td>
<td>35</td>
<td>14</td>
<td>0.7629</td>
<td>0.84</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fig6p31</td>
<td>6</td>
<td>10</td>
<td>1</td>
<td>0.6250</td>
<td>1.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fig6p33</td>
<td>7</td>
<td>11</td>
<td>1</td>
<td>0.6667</td>
<td>0.90</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fig6p34</td>
<td>17</td>
<td>24</td>
<td>12</td>
<td>0.5785</td>
<td>0.93</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fig6p38</td>
<td>13</td>
<td>14</td>
<td>4</td>
<td>0.6212</td>
<td>0.77</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fig6p39</td>
<td>12</td>
<td>12</td>
<td>5</td>
<td>0.8986</td>
<td>0.89</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fig6p42</td>
<td>7</td>
<td>18</td>
<td>4</td>
<td>0.2284</td>
<td>0.74</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RelProc</td>
<td>21</td>
<td>28</td>
<td>12</td>
<td>0.7143</td>
<td>0.73</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BPIC2012</td>
<td>15</td>
<td>22</td>
<td>15</td>
<td>0.6195</td>
<td>0.79</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1. Comparison of the precision in selected process models discovered with Inductive Miner and then Unrolled. The simplicity of the processes is also depicted with the number of places \( P \), transitions \( T \) and silent activities \( \tau \) in the discovered Petri Nets. For some cases the unroll is not possible without sacrificing fitness.

Table 1 contains the precision obtained with the process model discovered with Inductive Miner (IM), the precision obtained after unrolling these process models and precision obtained by PNSimpl [5]. We have used the alignment-based precision metric [2] for this evaluation. 7 out of 15 processes do not show any improvements with our technique since it was not possible to perform an unroll without losing fitness. For the BPI Challenge 2012 log, it was also not possible to perform an unrolling without losing fitness. None of the iterative subprocesses was repeated a multiple of \( k \) times for any \( k \). Nevertheless, for this dataset we followed another strategy: We choose \( k \) in order to minimize the loss in fitness. In this particular case, after performing a 2-unrolling on the activity \( \text{Calling to add missing information to the application} \), 9% of the traces cannot be replayed by the unrolled process model, with a minimal impact on fitness, but its precision increases 5%.

Results indicate that precision gain is similar with both techniques, provided that unrolling is possible. Nevertheless, both approaches treat the initial process model differently: Our approach enhances the expressive power of the initial process, whilst PNSimpl rediscover the process after each label split and, hence, the final process might be significantly different. Besides, the ability of perform-
ing unrolls (by losing fitness) enables us to highlight interesting properties of the process. For instance, loop unrolling allowed us to check that the financing organization of the BPI Challenge 2012 usually had to call customers twice for getting the necessary information. Is there any reason one attempt is not enough?

In terms of complexity, the technique of this paper may be a light alternative for methods like [5], which require to iteratively apply agglomerative clustering for special sets in the state-space representation of the event log.

5 Conclusion

In this paper, we presented a method for improving the precision of structural subprocesses based on explicitly repeating iterative subprocesses and pruning unused constructs and activities. We have shown that this approach is applicable to simulations of real-life processes, and also it is applicable to real-life scenarios. The presented approach is the first step on considering the unrolling of iterative processes. Results in Table 1 show several examples of how unrolling improve the precision of the process models, with minimal impact on their complexity. Nevertheless, bigger process models might be more difficult to understand and, hence, it remains to conduct expert reviews on readability and understandability of process models after unrolling. Besides, we have experienced on some datasets that some iterative processes can be explained as a few iterations are used for initialization, and then the real loop starts. We would also like to study how the $k$-unroll operation affects the precision of the process model for a particular precision metric. In particular, is it possible to establish a lower bound on the increase of the precision?

Acknowledgment

This work is funded by Secretaria de Universitats i Recerca of Generalitat de Catalunya, under the Industrial Doctorate Program 2013DI062, and the Spanish Ministry for Economy and Competitiveness, the European Union (FEDER funds) under grant COMMAS (ref. TIN2013-46181-C2-1-R).

References


Clustering Software Developer Repository Accesses with the Cophenetic Distance

David Sánchez-Charles¹, Josep Carmona², Victor Muntés-Mulero¹, Marc Solè Simo¹

¹ CA Strategic Research, CA Technologies, Spain
{David.Sanchez,Victor.Muntes,Marc.Sole}@ca.com
² Universitat Politècnica de Catalunya, Spain
jcarmona@cs.upc.edu

Abstract. In this paper we report a case study on the use of the cophenetic distance for clustering software developer behavior accessing to a repository. This distance was recently proposed for the comparison of tree-based process models. We show how hierarchical clustering techniques over the cophenetic distance are capable of detecting homogeneous clusters corresponding to well-defined software developer roles. The techniques have been evaluated on real-life data from a software development project, and the quality of the clusters over the cophenetic distance is compared with similar techniques in the literature.

1 Introduction

Globalization of large enterprises is encouraging the industry to move towards Software as a Service (SaaS) solutions and communicate in digital platforms. One of the key benefits of these SaaS solutions is that any employee can access it whenever and wherever they can, and with limited resources such as personal laptops or mobile phones. Besides, these technologies enable for decentralized collaboration with peers in completely different time zones, providing diversity and flexibility in the workplace.

In the context of software development, Apache Subversion (SVN) and GitHub are two software versioning and revision control systems that allow software developers to collaborate in the maintenance and development of software, by monitoring changes in files such as source code, web pages, and documentation. Those systems have a record of actions performed by users, in order to increase visibility of developers or in case some changes need to be reverted.

The objective of this paper is to test if we are able to recover the original function (or role) of some workers of an organization, based on the actions performed in the software repository. Instead of directly analyzing sequences of actions, we propose to summarize them in the form of a process model. I.e. we will assume that a process model is capable of representing the behavior of the individuals in the platform. On the long-run, this is the first step to assess if one is able to gain some knowledge about individuals by analyzing, and comparing, how they behave in a digital platform.

In case of success, the techniques of this paper open the door for interesting applications, ranging from profiling of user behavior, detection of outlier behavior that may be suspicious of malware or fraud, user interfaces and processes that adapt to the behavior of the user, and, in general, other scenarios in which we want to predict some
attribute of the process owner. It will be of particular interest in crowdsourcing projects, in which organizations outsource particular internal processes to large corpora of virtual workers across the globe, for allowing organization to understand which type of workers are contributing to their problem resolution.

2 Methodology

In this paper we will try to group users with a similar role by analyzing the actions that they performed in a digital platform. The following diagram summarizes the approach we followed in this paper:

```
Process Discovery → Process Model Comparison → Clustering
```

First, we preprocess the actions performed by the users and apply a process discovery technique over them. Following the process discovery, we will apply a process model comparison for measuring the pairwise dissimilarity of all process models and, implicitly, the dissimilarity of the human behavior that they represent. Finally, a clustering technique will be applied in order to discover groups of users with similar process models.

Section 2.1 briefly discusses the discovery of the process models that represents the behavior of the users. Section 2.2 covers the considered similarity metrics on this case study, and then Section 2.3 briefly mentions some clustering techniques appropriated for this scenario. Later in Section 3, we will evaluate such metrics and clustering techniques by applying them on a real-scenario in which the roles of the individuals are already known.

2.1 Process Discovery

The first step would be to generate a representation of the behavior of users in the digital platform. For this we have chosen to discover process models that summarizes the sequences of actions performed by users. Analyzing behavior over process models, instead of directly on the sequences, allow us to not focus on the specifics (such as, for instance, activity $A$ has been repeated 5 times) but the general overview of how the user behaved (following the example, a process model simply states that activity $A$ is usually repeated several times, whilst the number of times is not relevant).

Some sort of preprocessing may be needed in order to generate a process model. In particular, we may need to clean event names or ensure that traces represent independent runs of the underlying process. In our case, traces must compromise actions in a single session, with a unique purpose. In case this information is not provided, some heuristics must be applied in order to artificially split those sessions. For example, a login event may clearly separate two different sessions.

As for the process discovery itself, we will use the infrequent Inductive Miner [7]. This algorithm provides a good generalization of process models, can be efficiently applied to large event logs, and its output is a process tree – as required by some distances used in Section 2.2.
2.2 Process Model Comparison

The state of the art techniques for comparing process models can be naively split into structural and behavioral. In the former, process models are considered as labeled graphs and the comparison is regarded as edit operations over their edges, nodes or both. In contrast, behavioral techniques focus at the comparison of the execution semantics of the compared models.

If we focus on the case of structural comparison of process models, techniques based on graph edit operations have been defined [5,3,4,12]. In particular, we will consider the graph edit distance as defined in [5] for representing the group of structural similarity metrics. This distance counts the number of modifications (addition or removal of nodes and edges) must be performed in order to transform one process model into the other.

Analogously to structural techniques, behavioral ones compare how activities are related to each other. A behavioral profile of a process is a representation of the process model as an \( n \times n \) matrix, where \( n \) is the number of tasks in the process [11]. An element \( w_{i,j} \) in the matrix states the behavioral relation between the activity represented by the row \( i \) and the column \( j \), which can be causality (typically depicted by \( > \)), reverse causality (\( < \)), mutually exclusive (\( \lor \)) or co-occurrence (\( \land \)) in the case of causal behavioral profiles. Behavioral profiles provides one mechanism for comparing the behavior of two process models, by counting the number of cells in which the two processes do not have the same behavioral relation.

![Behavioral Profile Diagram](image)

**Fig. 1.** Example of a Casual Behavioral Profile for a process model. Activity \( A \) is concurrently executed (\( \land \)) with activities \( B \) and \( C \), whilst activities \( B \) and \( C \) cannot be executed both in the same run as they are mutually exclusive (\( \lor \)). Finally, activity \( D \) is a consequence (\( > \)) of activities \( A \), \( B \) and \( C \).

In [9], we showed that the boundary between structural and behavioral metrics is not very clear, and some metrics capture behavioral differences even though they are purely defined in an structured manner. The Cophenetic Distance fall in this fuzzy boundary, and we will also consider it in this study.
In the rest of this section, we provide an informal explanation of the Cophenetic distance, whilst the rest of distances are well-known in the literature. The use of this metric limits the scope of the study to process trees in which activities are not duplicated. Nevertheless, the inductive miner algorithm already provides such type of process models, and such constraint also makes feasible the computation time of the aforementioned techniques.

**Cophenetic Distance over Process Trees** A process tree is a labeled rooted tree $T$ in which activities are represented as leaves of the tree and internal nodes describe the control-flow of the process. For the sake of simplicity, we will label internal labels as $\textbf{OR}^3$, $\textbf{AND}$, $\textbf{SEQ}$ and $\textbf{LOOP}$ to represent the usual behavioural structures in a process model. We will also denote by **gateways** to these internal nodes, following the BPMN nomenclature. Children of a SEQ gateway are ordered in order to represent the sequential ordering of the subprocesses they represent. We allow silent activities by labeling them as $\emptyset$. Figure 2 depicts an example of a sequential process with a optional branch and two concurrent activities. On the right, the same process is represented as a process tree.

![Diagram of a process tree](image)

**Fig. 2.** Example of a structured process and its translation into a process tree. Activities $B$ and $C$ are concurrently executed, but are completely optional. The process always starts with activity $A$ and ends with activity $D$.

In any process tree $T$, the deepest common ancestor of two nodes $u$ and $v$—denoted by $[u, v]_T$—holds the direct causal relationship of the two nodes, and the depth of the common ancestor denotes the complexity of the process structure up to this behavioural decision. The depth of $[u, v]_T$ is known as **Cophenetic value**, simply denoted by $\text{Depth}_T$.

---

3 Following the semantics of block-structured models in [1], only exclusive ORs are modeled.
(\([u, v]\)\), and the Cophenetic vector is the collection of such Cophenetic values for every possible pair of nodes \(u\) and \(v\). Authors in [2] show that the Cophenetic vectors are enough to discern the structure of a certain class of labeled trees, which includes process trees without activity repetitions. And therefore, we could define a structural distance between process trees based on their cophenetic values.

**Definition 1.** Let \(T\) and \(T'\) be two process trees, and \(S\) the set of activities of the two trees. Their cophenetic distance is

\[
d_{\varphi}(T, T') = \sum_{i, j \in S} |\text{Depth}_T([i, j]_T) - \text{Depth}_{T'}([i, j]_{T'})|
\]

For the sake of simplicity \(\text{Depth}([i, j]_T)\) is zero if either \(i\) or \(j\) are not present in the process tree \(T\).

Unfortunately, cophenetic vectors are not enough for determining behavioral similarity: for instance, Figure 3 depicts two similar processes where two internal nodes have been interchanged, but share the same cophenetic vector due to having identical structures. In [9], we presented a new approach to compare process trees using the cophenetic distance that, by modifying the notion of tree depth, enables us to overcome such issue and leverages the traditional structural comparison with the behavioural information hard-coded into the new depth. Let’s take the left model of Figure 4 to illustrate some of the rules defined in [9]. Activities \(A\) and \(B\) originally had depth 3, but 2.5 considering the proposed depth definition: the depth of their parent, which is 2, plus 0.5 given the dichotomy of the exclusive choice they are representing. The \(\text{AND}\) gateway was originally at depth 2 since it was a direct child of the root, whilst we are now positioning it at depth 3.5 as if it was a direct consequence of the two activities \(A\) and \(B\). These consecutive depths are triggered by the behavioral function of its parent node – a sequential construct. One can check [9] for more details on how this new depth is defined.

### 2.3 Clustering of Process Models

At this point, we already have a process model summarizing the behavior of each individual as explained in section 2.1. We have computed a matrix \(D\) of distances between
all process models, i.e. cell $D_{i,j}$ includes the distance between the process models of users $i$ and $j$. This distance may be defined by the graph edit distance [5], differences in their causal behavioral profiles [11] or the cophenetic distance [9] as briefly discussed in section 2.2.

In this final step, our objective is to create groups of individuals that are close to each other with respect to the distance matrix $D$. The standard de-facto technique for clustering based on a distance matrix is Hierarchical clustering.

**Hierarchical clustering** Hierarchical clustering [8] follows a bottom-up approach, in which every individual starts within their own cluster and then iterates by merging the two most similar clusters into a bigger cluster. One tackles the issue of finding the two most similar clusters by averaging the distances of the individuals within the clusters. The only drawback of this technique is that the number of groups must be manually fixed. In our experiments, we will run the hierarchical clustering for all the possible number of groups.

### 3 Evaluation

In this section, we test the methodology explained in Section 2 with a real industrial dataset. First, we describe the provided behavioural data of 200 individuals and their organizational roles in the company. Process discovery is used to summarize the behavioural data from each individual, and dissimilarities between those processes are meant to measure differences in the behaviour of the individuals. Then, we test some clustering approaches with three different similarity metrics to measure how good they approximate the original organizational rules.

#### 3.1 Framework of the evaluation

Apache Subversion (SVN) is a software versioning and revision control system. Software developers use SVN software to collaborate in the maintenance and development of software, by monitoring changes in files such as source code, web pages and documentation. All accesses to a SVN repository are done through HTTP/S, as specified in the WebDAV/DeltaV protocol. It turns out [10] that those read and write requests over HTTP/S can be translated to human-friendly SVN commands such as `svn update` or `svn

---

**Fig. 4.** Example of process trees and their cophenetic vector (in matrix representation), using the depth described in [9]. For simplicity, we included node’s depth as a subscript of the label.
Continuing the work done by Li Sun et al. [10], we model the behavior of developers by using process discovery techniques. First, SVN commands are retrieved from the system and considered as events of a system that represents the developer, and then a trace is defined as all commands executed during a complete business day. As already mentioned in Section 2.1, we discover Process Trees using the default settings of the Inductive Miner Plugin (ProM 6.5.1).

This industrial dataset contains all the accesses of more than 200 individuals to one repository of CA Technologies in production for three years. After pruning users with few accesses to the repository, 83 individuals were kept in the study and their organizational roles were retrieved at the end of the monitoring phase. In particular, 37 Forward Engineering, 19 Quality Assurance Engineers, 16 Sustaining Engineer, 5 Support, 2 Services, 1 SWAT Engineer, 1 Infrastructure, 1 Technical Writers. The following list summarizes the responsibilities for each role.

- **Forward Engineers** (R1) are in charge of the implementation of new features.
- **Quality Assurance Engineers** (R2) plan, run and design use cases or tests.
- **Sustaining Engineers** (R3) are in charge of solving defects, as well as ensuring that software successfully passes all tests.
- **SWAT engineers** (R4) are in charge of implementing custom integrations.
- **Support** (R5), **Services** (R6) and **Infrastructure Engineers** (R7) interact with internal and external customers with respect to defect detection and solution, software installation and configuration, and maintenance of the infrastructure of Software as a Service solutions provided by the company. **Support Engineers** might push some quick fixes into products.
- **Technical Writers** (R8) collaborate with Forward, Sustaining and Quality Assurance Engineers for creating helpful Knowledge Base and User Guides. Technical Writers are asked to use the source code repository to maintain different versions of the documentation.

Among all the engineers, and fairly distributed among roles, 9 individuals are Managers of a team. Besides, one agent is labeled as a bot, although the purpose of such bot is unknown to the authors of this paper. Notice that one has the possibility of advancing in their career and change to another department, and, therefore, some individuals might have been misclassified as their latest role. Infrastructure and Service Engineers are not supposed to access the repository in their usual pipeline and, therefore, might have been promoted during the project. Nevertheless, clustering may help us to deduce their original roles in the organization.

During the rest of the evaluation we plan to answer the following question in regard of this scenario:

- How good is clustering of process models for approximating the original role of the individuals?
- Which is the expected role of the bot? And what about the role of other anomalies?

### 3.2 Homogeneity of roles in process-based Clustering

In order to measure the quality of the clustering, we will use the purity as the metric for measuring the homogeneity of the discovered groups. Let $C = \{C_1, \ldots, C_m\}$ be a
clustering of the process models, and $R_i(C_j)$ be the number of individual in cluster $C_j$ with the role $R_i$, then the purity is defined as

$$\text{Purity}(C, R) = \frac{1}{\text{Number of processes}} \sum_{j \in C} \max_{R_i \in R_i(C_j)}$$

In other words, the purity computes accuracy as if we label all individuals inside a group with the most popular role inside it. In particular, very heterogeneous groups of individuals will lead to a poor purity.

Figure 5. The solid line depicts the evolution in the purity of the Cophenetic-based hierarchical clustering as the number of clusters increase, whilst the dashed (resp. dotted) line depicts the graph edit distance (resp. Behavioural Profiles). Two different experiments were performed for detecting individuals’ role and their status as managers.

Table 1 summarizes the precision and recall of the hierarchical clustering of Figure 5 when we set the number of clusters to 6. Again, we consider the classification as if the predicted label of all elements inside a cluster is the role of the more popular role inside the cluster. These results show again the capabilities of the Cophenetic distance to highlight the Manager role and provides, in general, better results for all roles, except the Sustaining Engineer. On the other hand, GED and Behavioral Profiles tend to group all individuals in a very big, and heterogeneous, cluster and, therefore, we obtain those roles with high recall but low precision (and those with high precision but very low recall).
As for the bad performance with respect to Sustaining Engineers, notice that responsibilities of the Sustaining (R2), Quality Assurance (R3), SWAT (R4) and Support engineers (R5) are all related to defects and bug fixing, and, therefore, they may share some common behaviour and practices. Besides, the number of Sustaining Engineers is slightly below the number of Quality Assurance Engineers, and, hence, it is more likely to label users as Quality Assurance Engineers in case of grouping them together. Table 2 summarizes the precision and recall for a clustering of 6 groups. Notice that precision and recall of the Forward Engineer category are not significantly affected in the case of the cophenetic distance, indicating the existence of groups with a strong presence of Forward Engineers. On the other hand, precision and recall are very affected in both GED and Behavioral Profiles cases. The results provided by the GED are an indication of one or more small groups groups of Forward Engineers (perfect precision, but low recall), and a big group in which half of the developers have a role in R2345 and the rest are Forward Engineers or other minor roles. As for behavioral profiles, results are slightly worse than the cophenetic distance, but still incapable of detecting the group of Managers.

We have run the same experiments using DBSCAN [6] as the clustering method. The key benefit of DBSCAN is that the number of clusters is not fixed prior to the clustering, as it defines clusters as groups of individuals that are densely together\(^4\). Unfortunately, results are significantly worse than the provided by the hierarchical clustering – with purity not surpassing 0.5 across several hyperparameter of the DBSCAN algorithm.

### Table 1. Accuracy and Recall for each of the roles in the organization by considering a hierarchical clustering with 6 groups. In some cases, none of the groups had enough representation of a role.

<table>
<thead>
<tr>
<th>Role</th>
<th>Cophenetic</th>
<th>GED</th>
<th>Behavioral Profiles</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Precision</td>
<td>Recall</td>
<td>Precision</td>
</tr>
<tr>
<td>Manager</td>
<td>0.71</td>
<td>0.55</td>
<td>No indiv.</td>
</tr>
<tr>
<td>Forward Engineer</td>
<td>0.64</td>
<td>0.76</td>
<td>0.46</td>
</tr>
<tr>
<td>Sustaining Engineer</td>
<td>0.00</td>
<td>0.00</td>
<td>0.57</td>
</tr>
<tr>
<td>Quality Assurance</td>
<td>0.34</td>
<td>0.57</td>
<td>No indiv.</td>
</tr>
<tr>
<td>Support</td>
<td>0.50</td>
<td>0.40</td>
<td>No individual was labelled</td>
</tr>
<tr>
<td>Others</td>
<td></td>
<td></td>
<td>No individual was labelled</td>
</tr>
</tbody>
</table>

\(^4\) I.e. for every process model in the cluster, there must be at least \(k\) process models at distance less or equal than \(d\). Both \(k\) and \(d\) are manually fixed.

---

### Inducing the real role of outliers

Some role anomalies were present in the dataset. For instance, two individuals were classified as Service Engineers (R6) although accessing to the source code repository is not part of their responsibilities. As we have already mentioned, the role data was obtained during the finalization of the project and, hence, the worker may have changed from one department to another. In this case, one service engineer (R6) is more close to Quality Assurance Engineers (R3), and the other is close to a group of Forward Engineers (R1). The three distances are consistent with
Table 2. Precision and Recall for each of the roles in the organization by considering a hierarchical clustering with 6 groups after merging Sustaining, Quality Assurance, SWAT and Support Engineers into a unique role R2345. In some cases, none of the groups had enough representation of a role.

<table>
<thead>
<tr>
<th>Role</th>
<th>Cophenetic Precision</th>
<th>Cophenetic Recall</th>
<th>GED Precision</th>
<th>GED Recall</th>
<th>Behavioral Profiles Precision</th>
<th>Behavioral Profiles Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manager</td>
<td>0.71</td>
<td>0.55</td>
<td></td>
<td></td>
<td>No individual was labelled</td>
<td></td>
</tr>
<tr>
<td>Forward Engineer (R1)</td>
<td>0.60</td>
<td>0.78</td>
<td>1.00</td>
<td>0.05</td>
<td>0.65</td>
<td>0.35</td>
</tr>
<tr>
<td>R2345</td>
<td>0.72</td>
<td>0.63</td>
<td>0.50</td>
<td>1.00</td>
<td>0.56</td>
<td>0.85</td>
</tr>
<tr>
<td>Others</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>No individual was labelled</td>
<td></td>
</tr>
</tbody>
</table>

these results. With respect to the Infrastructure Engineer (R7), the cophenetic distance and behavioural profiles map this user close to Sustaining Engineers (R2) whilst the graph edit distance relate him to Forward engineers (R1). Finally, with respect to the agent labeled as a BOT, the cophenetic and the graph edit distance group it with other Quality Assurance Engineers (R3). This might be a hint that the bot is indeed an automatic testing system. Nevertheless, Behavioural Profiles are less accurate and relate this agent close to a mixed group of Forward (R1), Sustaining (R2) and Quality Assurance engineers (R3).

4 Conclusions and Future Work

In this paper, we have applied the cophenetic distance for process models to recover groups of individuals with similar behaviours, and hence similar roles and responsibilities. Our approach is based on comparing the behaviour of process models discovered on real logs, instead of comparing them directly on the logs, allowing us to compare a generalization of the behaviour instead of falling into the specificity of traces. For instance, our approach allowed us to realize that a bot was working for a specific team, as this bot behaved as the other team members. We compared our cophenetic approach with three other process similarity metrics and we have seen that our approach consistently provides better role retrieval, as well as detecting a small group of individuals acting as Managers of a team.

As future work, we would like to investigate the possibility of discovering process models for each cluster such that the trace-fit within the individuals in the cluster is high, whilst significantly lower when applied to individuals outside the cluster. That would help in understanding the behavior of new users into the system.

Acknowledgements. This work is partially funded by Secretaria de Universitats i Recerca of Generalitat de Catalunya, under the Industrial Doctorate Program 2013DI062, and the Spanish Ministry for Economy and Competitiveness (MINECO), the European Union (FEDER funds) under grant COMMAS (ref. TIN2013-46181-C2-1-R).
References

Short Papers
Analysis of Energy-Efficient Buildings through Simulation and Formal Methods

Maryam Ehsanpour¹, Luciano Baresi ², Matteo Rossi ²
Ernesto Damiani³

¹ Dipartimento di Informatica, Universita degli Studi di Milano, Via Comelico 39/41,
20135 Milano, Italy
maryam.ehsanpour@unimi.it

² Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano
Piazza Leonardo da Vinci 32, 20133 Milano, Italy
{luciano.baresi,matteo.rossi}@polimi.it

³ Department of Electrical and Computer Engineering, Khalifa University
Al Zafranah, Abu Dhabi, United Arab Emirates
ernesto.damiani@kustar.ac.ae

Abstract. In the last few years, the increasing interest in energy saving has led to invest resources in many research fields, in order to develop new technologies to reduce energy consumption. One such field is energy management systems for buildings. The interactions between the system controlling the building and its environment are complicated by the fact that they involve important timing properties, which can be expressed through a variety of formalisms. This paper introduces a formal model, expressed in a temporal logic language, through which designers can evaluate the temporal behavior of buildings’ energy management systems. To this end, the formal model is analyzed through simulation, that is, through the generation of traces satisfying the temporal logic model.

Keywords: Data analysis, Energy efficient, Formal methods.

1 Introduction

Managing the technology that is used to control the various functions of modern buildings is a complex task. The U.S. Department of Energy estimates that buildings in industrialized countries account for around 40% of total energy consumption. Researchers have shown that savings in the order of 20-30% can be gained through careful energy management and control in buildings, even without changing the structure of the hardware configuration of their energy supply systems[1]. The problem of modeling and designing software control systems has been approached through different techniques, such as temporal logic languages[2], which are supported by a wide range of
techniques for model analysis and verification. This work aims to introduce a solution for describing the management and control of energy consumption in buildings through rules expressed in temporal logic. The architecture of the approach is described through a simple example, namely the control of Rooms provided with various appliances.

The rest of the paper is organized as follows. Section 2 introduces the simulation tools on which the approach is founded; Section 3 introduces the temporal logic language used for describing control rules; and Section 4 illustrates the case study which exemplifies the main characteristics of the proposed analysis method based on temporal logic rules. Section 5 concludes. The Appendix shows an example of execution trace produced through the Zot tool, which is used in this work to simulate temporal logic models.

2 Tool support

2.1 OpenModelica

A number of well-established techniques have been developed over the years to model physical phenomena, which constitute the environment of software systems. Typically, these phenomena are described through a continuous notion of time, using approaches that can be classified as either causal or acausal. In causal approaches, the model is decomposed into blocks which interact through input and output variables. The object-oriented language Modelica [3], instead, follows a acausal approach where each (physical) component is specified through a system of differential algebraic equations; connections between modules, which correspond to physical principles, are set by equating effort variables and by balancing flow variables. OpenModelica is an open-source simulation engine based on the Modelica language; it can be interfaced with the Zot tool described below to create closed-loop simulations of software systems and their environments [4].

2.2 Zot

Zot can perform formal verification of models described through temporal logic formulae. More precisely, let us call $S$ the model of the designed system, which corresponds to a set of temporal logic formulae. If $S$ is input to the Zot tool, the latter looks for an execution trace of the modelled system; that is, for each time instant and for each predicate appearing in the formulae of $S$, it looks for an assignment of values which make the formulae of $S$ true. If no such trace is found (i.e., if the model is unsatisfiable), then the model is contradictory, meaning that it contains some flaw that makes it unrealizable in practice.

1 http://openmodelica.org
2 github.org/fm-polimi/zot
3 The TRIO temporal logic language

In this section we provide a brief overview of the temporal logic, called TRIO [1], which has been used in this work to capture the rules governing the energy management system. TRIO allows users to express timing properties of systems, including real-time ones. TRIO adopts a linear notion of time, and can be used to express properties over both discrete and continuous temporal domains. Table 1 presents some typical TRIO temporal operators, which are defined in terms of the basic Dist operator, where Dist(φ, d) means that formula φ holds at the instant exactly d time units from the current one.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Definition</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Futr(φ, d)</td>
<td>d &gt; 0 ∧ Dist(φ, d)</td>
<td>φ occurs d instants in the future</td>
</tr>
<tr>
<td>Past(φ, d)</td>
<td>d &gt; 0 ∧ Dist(φ, -d)</td>
<td>φ occurs d instants in the past</td>
</tr>
<tr>
<td>Alw(φ)</td>
<td>∀(Dist(φ, d))</td>
<td>φ always holds</td>
</tr>
<tr>
<td>Lasts(φ, d)</td>
<td>∀(0 &lt; t &lt; d → Futr(φ, t))</td>
<td>φ holds for the next d time units</td>
</tr>
<tr>
<td>Lasted(φ, d)</td>
<td>∀(0 &lt; t &lt; d → Past(φ, t))</td>
<td>φ helds for the last d time units</td>
</tr>
<tr>
<td>Until(φ, ψ)</td>
<td>∃(Futr(ψ, t) ∧ Lasts(φ, t))</td>
<td>ψ will occur and φ will hold until then</td>
</tr>
<tr>
<td>Since(φ, ψ)</td>
<td>∃(Past(ψ, t) ∧ Lasted(φ, t))</td>
<td>ψ occurred and φ held since then</td>
</tr>
</tbody>
</table>

4 Case Study: Rooms with Appliances

In this section we briefly describe how the energy management system is captured through the TRIO temporal logic language. In particular, we focus on an example of building whose rooms are provided with appliances whose energy consumption needs to be controlled. The model of the building is fed to the Zot tool to generate traces, which can then be analyzed to unearth anomalous behaviors. Fig. 1. shows the corresponding Modelica diagram describing the structure and the components of the analyzed system. The appliances considered are: 1. the Electric Boiler, whose desired temperature is 70°C; 2. the Washer whose desired temperature is 60°C and whose washing cycle lasts an hour; 3. The Bath, which can be used both as shower or as a bath; 4. The Oven, whose nominal power consumption is 1000 W; 5. The DishWasher, whose washing cycle is an hour and a half; 6. The Refrigerator, whose nominal power consumption is 90 W; 7. The HoodCookTop and the Stove, which are used when a user cooks with the stove and keeps the hood turned on.
These appliances are connected through three different networks: the hydraulic, the thermal, and the electric network.
In the rest of this section, we present the formulae capturing the behavior of the appliances.
For the **Electric Boiler**, if the water temperature is less than 70°C for 2 minutes, then the boiler is turned on (represented by predicate `boilerOn` being true):

\[
\text{Alw(}\text{Lasted(boiler\_water\_temp< boiler\_desired\_temp, 2))}\Rightarrow \text{boilerOn}).
\]

If the water temperature is higher than 70°C for 2 minutes, then the controller is turned off (i.e., “not boilerOn” holds):

\[
\text{Alw(}\text{Lasted(boiler\_water\_temp> boiler\_desired\_temp, 2))}\Rightarrow \neg \text{boilerOn}).
\]

The following formula constrains the difference between the current temperature and the one at the next minute when the boilers turned on; in particular, the temperature must increase (i.e., the difference is greater than 0), but by no more than 2 degrees (i.e., the difference is less than 3).

\[
\text{Alw(boilerOn}\Rightarrow \text{next(boiler\_water\_temp) - boiler\_water\_temp> 0} \land \text{next(boiler\_water\_temp) - boiler\_water\_temp< 3}).
\]

Also after the boiler is turned off, the temperature decreases by with a rate of at most 2 degrees per minute:

\[
\text{Alw}(\neg \text{boilerOn}\Rightarrow \text{next(boiler\_water\_temp) - boiler\_water\_temp< 1} \land \text{next(boiler\_water\_temp) - boiler\_water\_temp> -3}).
\]

The following formula constrains the range of the temperature to be between -20°C and 100°C:

\[
\text{Alw(boiler\_water\_temp< 101} \land \text{boiler\_water\_temp> -20})
\]

**Washer:**

The time washing cycle lasts60 min:

\[
\text{Alw(washer\_start}\Leftrightarrow \text{Futr(washer\_end, 60))}
\]

The washer is on between a “start” and an “end” command.
The washer cannot start again until the current cycle has finished.

\[ \text{Alw}(\text{washer}_\text{on} \Rightarrow \text{Since}_{\text{ei}}(\neg \text{washer}_\text{end}, \text{washer}_\text{start})) \]

**Dryer**:

The model of the dryer is very similar to the one of the washer.

The time spin/dryer cycle is 30 min.  \[ \text{Alw}(\text{dryer}_\text{start} \Rightarrow \text{Futr}_{\text{ei}}(\text{dryer}_\text{end}, 30)) \]

The dryer is on between a “start” and an “end” command.

\[ \text{Alw}(\text{dryer}_\text{on} \Rightarrow \text{Since}_{\text{ei}}(\neg \text{dryer}_\text{end}, \text{dryer}_\text{start})) \]

The dryer cannot not start again until the current cycle has finished.

\[ \text{Alw}(\text{dryer}_\text{start} \Rightarrow \text{Until}_{\text{ei}}(\neg \text{dryer}_\text{start}, \text{dryer}_\text{end})) \]

**Oven**:

If the door of the oven is opened for 2 minutes, the controller is on (i.e., it flashes):

\[ \text{Alw}(\text{Lasted}(\text{oven}_\text{door}_\text{open}, 2) \Rightarrow \text{oven}_\text{door}_\text{flash}) \]

**Dishwasher**:

The model of the dishwasher is also very similar to the one of the washer and of the dryer.

**Refrigerator**:

The model of the refrigerator is similar to the one for the oven; that is, if the refrigerator door is opened for 2 minutes, the controller is on (i.e., it flashes).

5 **Conclusion**

This paper introduces imulation-based mechanisms based on rules formalized in temporal logic, which can help improve the user’s understanding of modeled systems. The proposed approach can be used to model (complex) systems and describe their properties formally; it allows the user to simulate the behavior of the system and verify the satisfiability of the properties of interest. We exemplified our approach on a case study concerning the modeling of an energy management system for a building.

**References**

Appendix: Example of execution trace of the model

------ time 0 ------
BOILER_WATER_TEMP = 68.0
REF_DOOR_FLASH BOILERON
REF_DOOR_OPEN
------ time 1 ------
BOILER_WATER_TEMP = 70.0
BOILERON
------ time 2 ------
BOILER_WATER_TEMP = 71.0
------ time 3 ------
BOILER_WATER_TEMP = 69.0
REF_DOOR_FLASH
BOILERON
------ time 4 ------
**LOOP**
BOILER_WATER_TEMP = 70.0
------ time 5 ------
BOILER_WATER_TEMP = 68.0
REF_DOOR_OPEN
------ time 6 ------
BOILER_WATER_TEMP = 66.0
------ time 7 ------
BOILER_WATER_TEMP = 64.0
BOILERON
------ time 8 ------
BOILER_WATER_TEMP = 65.0
BOILERON
------ time 9 ------
BOILER_WATER_TEMP = 66.0
BOILERON
------ time 10 ------
BOILER_WATER_TEMP = 67.0
BOILERON
------ time 11 ------
BOILER_WATER_TEMP = 68.0
BOILERON
------ time 12 ------
BOILER_WATER_TEMP = 69.0
OVEN_DOOR_OPEN
BOILERON
------ time 13 ------
BOILER_WATER_TEMP = 70.0
BOILERON
------ time 14 ------
BOILER_WATER_TEMP = 71.0

BOILERON
------ time 15 ------
BOILER_WATER_TEMP = 72.0
OVEN_DOOR_OPEN

BOILERON
------ time 16 ------
BOILER_WATER_TEMP = 73.0
OVEN_DOOR_OPEN

BOILERON
------ time 17 ------
OVEN_DOOR_FLASH
BOILER_WATER_TEMP = 73.0
REF_DOOR_FLASH

BOILERON
------ time 18 ------
BOILER_WATER_TEMP = 71.0

BOILERON
------ time 19 ------
BOILER_WATER_TEMP = 70.0
OVEN_DOOR_OPEN
REF_DOOR_OPEN

BOILERON
------ time 20 ------
BOILER_WATER_TEMP = 69.0
REF_DOOR_FLASH

BOILERON
------ end ------
Analysis of coordinating activities in Collaborative Working Environments

Muhammad Muneeb Kiani, Abdul Rafay Awan
University of Milan, Milan Italy, Teradata, Islamabad Pakistan
Muhammad.kiani@unimi.it, abdul.rafay@teradata.com

Abstract. Collaborative Working Environments (CWE) are widely used for effective collaboration among users. A CWE includes various tools and methodologies to support analysis of coordinating activities. Users within a CWE widely utilize textual means of collaboration and communication. An effective analysis of this textual collaboration can help in improving overall quality of collaborating activities and monitoring of the CWE. In textual analysis text is analyzed within certain context. Existing semi-automated techniques which are based on lexical, syntactic, semantic and other analysis approaches can be utilized with addition of customized automated classifier to cater needs of analyzing coordinating activities in collaborative working within a specific context. In proposed framework, natural language processing, opinion mining, lexicon based approaches will serve as processors of the framework.

Keywords: Text Analysis, Collaborative Working Environments

1 Introduction

Advancements in technology made possible interactive communication with computer systems and among other users. Further down the road with the help of these new technologies paradigm of collaborative working emerged. Set of tools that supports notion of collaborative working are known as Collaborative Working Environment (CWE) [1]. Part of coordinating activities in collaborative working environment includes textual communication. Though there are various techniques available for monitoring collaborative working but monitoring textual communication is still a complex task.

NLP (Natural Language Processing) can offer relevant support for the automatic classification of actions. In recent years researchers developed various mechanisms and algorithms for analyzing text. The term text analysis refers to the tasks that are performed to interpolate the facts and figures to augment the decision making and predicting future trends. There are two categories of text analysis; first category is the analysis of structured data that is performed on the data warehouse of an organization to find out different statistics of a business whereas the second category is the analysis
of unstructured data i.e. web logs, audios, videos, etc. to predict the market trend and what are the reasons for the failure of a particular product etc.

Text analysis techniques can be useful in monitoring coordinative activities within a CWE, understanding coordination among users in a given CWE and context can greatly enhance overall effectiveness of the systems. Though textual analysis techniques mentioned earlier are very mature, but they require further customization in context of CWEs for an accurate analysis.

In this paper a model is presented to analyze the collaboration within a certain context by using existing text analysis techniques to augment the CWE.

2 Background

The increasing degree of connectivity has given rise to the collaborative environments. Governments and corporations have adapted to networked collaborative environments to deliver their services. Managing the ever changing dynamics of collaborative environments and putting in place effective monitoring processes has become an important competency parameter. In order to determine the quality index of text based collaborative environments use of NLP is the inherent choice. Lexicon based techniques were used to analyze the activity model as proposed by the activity theory to analyze and identify the cognitive advantages of joint activity [2].

As mentioned earlier in section 1, CWE utilizes textual communication means for coordinating activities. These activities are performed with the help of rich text editors, group chat messages, emails and other means. This involves a lot of textual data, and effective understanding and monitoring of this data can greatly help in improving systems and overall activities from various aspects. Textual analysis is one way to understand this information. Following is a small brief of various textual analysis methods.

Some research work has been done in other languages for word sense disambiguation [3]. Basing on the work of WordNet, Esuli and Sebastiani introduced another library named as SentiWordNet for the purpose of opinion mining. SentiWordNet utilizes lexical basis of WordNet and assigns certain value to different words in terms of positivity, negativity and neutrality which as result help in determining overall mood of a textual data [4].

3 Need for Analysis of coordinating activities in CWE

Lyk et al explained role of monitoring in a CWE in various stages [5]. Great deal of work has been done and various open source libraries of text mining, classifying and analyzing text have been made available in recent years. Opinion mining or sentiment analysis determines overall mood of the textual data focusing on particular set of activities within a data set [6]. Isabella et al. analyzed coordinating activities against a predefined set of parameters [7]. But classification of individual items has been done
manually without a particular rule set so it is not possible to scale up or use this approach in similar scenarios. Secondly slicing of text has been done based on time stamp and later rectified manually with human intervention which is another bottle neck in this approach. 

There is a need to better understand set of activities within context of a collaborative working environment which involves overall goal of collaboration such as brainstorming, surveys, coordinating activities, number of participant, timing etc. In order to analyze coordinating activities in a CWE exiting text analysis techniques can be utilize.

4 Proposed methodology

In proposed methodology coordinating activities of users will be recorded. For this purpose Innovation Factory [8] CWE will be used as it provides event logs of the coordination. Various textual analysis techniques will be tested and one that fits best for the purpose will be adopted. [7] Manually labeled a set of coordinating activities as shown in the Table 1. This data along with manually labeled data from further experiments will be used to evaluate the output of analysis.

<table>
<thead>
<tr>
<th>Type</th>
<th>Originator</th>
<th>Text Token</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group-Chat</td>
<td>Samo Rumez</td>
<td>where are you all?</td>
<td>Situation Request</td>
</tr>
<tr>
<td>Group-Chat</td>
<td>Samo Rumez</td>
<td>no, i think Vesna is eating</td>
<td>Situation Update</td>
</tr>
<tr>
<td>Group-Chat</td>
<td>Vesna Paulic</td>
<td>ok guys lets start</td>
<td>Plan Propose</td>
</tr>
<tr>
<td>Text</td>
<td>Nikolaj Potočnik</td>
<td>we can open all tv channels for one week before</td>
<td>Lay outing</td>
</tr>
<tr>
<td>Poll Question</td>
<td>Primož Klasinc</td>
<td>Campaign should be about</td>
<td>Information/knowledge Request</td>
</tr>
<tr>
<td>Poll Vote</td>
<td>Nikolaj Potočnik</td>
<td>Standalone Mobia</td>
<td>Information/knowledge Provision</td>
</tr>
</tbody>
</table>

Table 1. Short Snapshot of Manually labelled Data

Architecture of proposed methodology comprises of three main sections, input, output and processor as depicted Fig.1 Input section provides Information about Classes; currently we have assumed four collaborating classes that are Query (sub class counter query), Opinion, Agreement, Argument (sub class counter argument).
More classes can be elicited depending upon the context such as classes defined by [7] includes (idea) generation, agreement, disagreement, neutral and coordination. Second input includes raw data set which should also include other value added information for comprehensive analysis such as timestamps, number of users, their input text, overall topics of discussions. Third and last input is set of quality metrics which helps analysis to assign weight to various activities based on type of context for example quality metrics will be different in case of a brainstorming, group discussion session than that of a question answer session or survey.

![Overall Architecture of Proposed Methodology](image)

Figure 1: Overall Architecture of Proposed Methodology

Second section includes tokenization rule engine which determine how text should be sliced for analysis purposes, it necessary to correctly tokenize set of text for correct semantic linking. Tokenizing for finding queries is easy but finding other elements is complex set. Determining a generic rule engine requires extensive evaluation in various settings.

As discussed earlier rule of classifier is to link tokenized text for creating meaningful information, in case of coordinating activities larger set of text are required to be classified to correctly determine their relevant classes. There are various libraries available which provides support text analysis such as dandelion which can be used for analysis [8]. Dandelion analyzes text with respect to context and also provides API for further customization and allows extraction of various kind of information. Last component of processer comprises of various environment variable such as time stamps, user participation, topic of discussion and related these variables with textual information in order to determining overall quality of entire activity.

Output section is set of reports which are produced after processing raw data and contextual information.
5 Conclusion and Future Work

Proposed methodology provides basis for developing a comprehensive framework and tool support for CWE. For the labeling of data, crowdsourcing tool will be developed. Results will be evaluated to mitigate the under-fitting or over-fitting of classifiers that can create false positives. Once the classifier is trained and tested against the raw data, in the third step it will be incorporated in a CWE such as Innovation Factory [8] analysis and monitoring framework.

References

Categorizing Identified Deviations for Auditing

Marzie Hosseinpour and Mieke Jans

Hasselt University, Agoralaan Gebouw D, 3590 Diepenbeek, Belgium

Abstract. Currently, financial statements auditors perform the tests of controls based on sampling. However, when using a sampling approach, information is lost. To counter this drawback, data analytics has been applied as a method for auditors to provide assurance while using all data. Specifically for testing controls, the potential of process mining has been explained in literature. Indeed, conformance checking can be used to compare real process executions with a normative model. However, the outcome of current conformance checking techniques is too vast for an auditor to inspect further. The identified deviations are at an atomic level (skipped and inserted tasks) and there is no feasible approach to gain a quick overview of the deviations. In this paper, we propose an approach to categorize deviations, which enables auditors to quickly gain an overview of different types of existing deviations along with their frequencies. Categorizing deviating process instances can also give an insight for assessing the risk at case level.

Keywords: Deviation Identification, Financial Statements Auditing, Process Mining, Conformance Checking, Risk Assessment

1 Introduction

The objective of financial audit practice, for auditors, is to express their opinion on correctness and fairness of organizations' financial statements. To achieve this goal, auditors test business process’s controls effectiveness and perform substantive tests of balances and accounts which impact the financial reporting. Test of process controls should be done before investigation of balance sheets and accounts. The rationale is that if the control settings are rigid, there is reasonable assurance, for the auditor, to rely on the organization’s internal controls. If the results of tests of controls are not satisfactory, more substantive evidence should be gathered. That means more time and effort should be dedicated while testing the balance sheets and statements. To test the controls, as required by Audit Standard No.5 [2] and ISA 315 [1] a general approach is to first collect information on the process by interviewing process experts and by studying the normative model (or creating one, in case it does not exist). After a general understanding of the normative model, auditors test the effectiveness of the associated control setting. Currently, this is tested by taking a sample of process executions. The sample is compared with the business model manually to check the conformity of the selected cases. If there are no deviating cases among this
selection, the control settings are assumed to be reliable. Otherwise, when deviating cases are discovered they should be reported. However, before reporting, the deviations should be studied by auditors and further discussed with process experts. The reason is that some of the deviating cases can be 'cleared' due to some implicit or explicit exceptional rules and be labeled as normal cases. The deviations which have a higher risk level should be prioritized for follow-up. Investigating deviations in such a way is a roughly feasible task owing to the currently used sampling approach. However, some information is lost and results may be inaccurate using sampling.

The advent of process mining techniques [5], in the last decade, can be promising for auditing, not in the least because of its full-population testing ([8,9]). By applying a conformance checking technique (such as [3,4,6,10,11]), the entire set of real process executions (aka event log) can be compared with the process model to distinguish deviating cases from normal cases. Although these techniques are able to locate the root cause of each deviation, for auditing purpose there are some shortcomings. First of all, the output of detected deviations is too immense for a full follow-up. There are too many variants of deviations. This is because the normative model does not cover all possible exceptional or flexible behavior of processes. The second problem is that almost all of the conformance checking techniques discover deviations in a very atomic level, which is not pragmatic to work with. For example, consider a procurement process with the model <Create PO, Sign, Release, IR, GR, Pay> where IR (Invoice Receipt) and GR (Goods Receipt) are concurrent activities and can change order. Take <Create PO, IR, Release, Pay> as an executed trace. If we check the conformity of this trace manually, we would intuitively notice that the activities Sign and GR are missing in this execution and Release and IR have changed order (since IR is expected to occur after Release, based on the model). By giving the model and the trace to the mostly employed conformance checking tool [6], the output will be as follow: Skipped(Sign), Skipped(Release), Inserted(Release), Skipped(GR). In a real event log, with thousands different cases, this leaves auditors with hundreds combination of low-level deviations, isolated from the variants and the context where they took place.

The question is how the idea of conformance checking (comparing a log with a model) can be made actionable for auditors. The approach we consider in this paper is to create a different layer of deviations that would allow us to categorize those deviations in sets that are meaningful and manageable for an auditor. Concretely, we would like to develop (or alter) a conformance checking algorithm that identifies deviations in such a way that i) they will be meaningful for auditors, ii) gives them an insight of different types of existing deviations and their frequencies in the executed behavior, and iii) helps auditors to prioritize deviations based on their risk level.

In the remainder of this paper, we propose an efficient interpretation of deviations in section 2, which can answer the above question. In section 3, we explain how we will implement the idea and we conclude in section 4.
2 Proposed Approach

To address the problem of too fine-grained deviation types (skipped and inserted) mentioned in the previous section, we define a set of six deviation types that we presume would be meaningful to an auditor, even without the context of the complete trace. These deviation types interpret deviations at a higher level, while keeping the location and root cause of the mismatches in each case. The six types that we propose are: missing a sequence, existence of an extra sequence, loop on a sequence, repetition of a sequence, swapping two sequences, replacing one sequence by another sequence. Note that a sequence can contain one or more activities. These proposed types are explained in table 1 due to lack of space. For each deviation type an example is provided, based on the procurement model presented in section 1.

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Missing a sequence</td>
<td>A sequence of events that should have taken place, is not executed</td>
<td>&lt;Create PO, Release, IR, GR, Pay&gt; Sign is missing</td>
</tr>
<tr>
<td>2 Existence of extra sequence</td>
<td>A sequence of events is executed that it was not designed</td>
<td>&lt;Create PO, Sign, Change Line, Release, IR, GR, Pay&gt;) Extra event Change Line exists</td>
</tr>
<tr>
<td>3 Loop on a sequence</td>
<td>A sequence of events is repeated while only a single occurrence was designed</td>
<td>&lt;Create PO, Sign, Release, IR, GR, Pay&gt;) Loop on Pay</td>
</tr>
<tr>
<td>4 Repetition of a sequence</td>
<td>An executed sequence is repeated later in another part of trace</td>
<td>&lt;Create PO, Sign, Release, IR, GR, Sign, Pay&gt;) Repetition of Sign after GR</td>
</tr>
<tr>
<td>5 Swapping two sequences</td>
<td>Two sequences have changed their order</td>
<td>&lt;Create PO, IR, Sign, Release, GR, Pay&gt;)</td>
</tr>
<tr>
<td>6 Replacing one sequence by another sequence</td>
<td>A sequence takes place instead of another missing sequence.</td>
<td>&lt;Create PO, Sign, Pay, IR, GR, Pay&gt; Release is replaced by Pay</td>
</tr>
</tbody>
</table>

Table 1: Deviation types from control-flow perspective with description and an example from procurement process explained in section 1 for each deviation type

The idea is to develop an algorithm that provides auditors with an overview of deviations according to their categories. Next, it should be feasible to drill down to the sequences that were subject to the deviations. For instance, the deviating trace <Create PO, IR, Release, Pay> will be described as follow: Missing an event (with two sub-categories: Sign is missed and GR is missed) and Swapping events (Release and IR are swapped). On a log level, for example, this could
give: "This logs shows 500 times a "Repetition of a sequence", this is stemming from 150 repetition of <Sign, Release>, 150 repetition of IR, and 200 repetition of GR." which describes existing deviations in the log in categories along with their frequencies.

Therefore, the contribution of this paper is to interpret deviations using these types, which enables auditors to perceive different types of deviations and possibly related risk, as one sees them intuitively (rather than on an atomic level).

3 Methodology and Implementation

Before the development of the desired algorithm, we will perform a field research. The methodology is to interview auditors to test our proposed deviation types in their approach. The field research will be executed to gain insights in how complex deviation types might be interpreted by human experts (as opposed to our assumptions). Consider again, the deviating example in section 1, <Create PO, IR, Release, Pay>. The deviation type 'swapped Release and IR' can be interpreted in a different way like 'Release is postponed after IR' or even 'IR is advanced before Release'.

When various deviation types exist in one trace, the combination of them also can be interpreted differently. Avoiding different interpretation of deviations is important because it may lead to assigning wrong risk level to them. Hence, we believe performing the field work research is necessary before implementation of the idea. After the field research, we plan to build the algorithm with the desired requirements. Current known algorithms have already been investigated to what extend they could help in achieving our goal. The shortcoming of cost-based conformance checking tool, proposed by Adriansyah et al. [6] is discussed in section 1. The conformance technique proposed by Garcia et al. [7], to the best of our knowledge, is the only conformance checking tool which provides the deviations in natural language statements. The tool finds deviations in both model and event log. The output is suitable to improve the model or see what types of deviations exist in the event log in general. Nevertheless, their approach is not fully compatible with auditing purpose of testing the controls. The reason is that it does not have the means for finding which cases or traces cause the discovered deviations. Hence, one is not able to find the deviating cases or even the frequency of each deviation type. After the field research, between these two techniques, we will choose the one which is closer to our objective of capturing and categorizing deviations and will adapt it to accomplish our goal.

4 Conclusion and Future Work

This paper introduces the idea of developing a technique for organizing the identified deviations in event logs into certain categories. A set of six different deviation types are proposed to enable auditors to gain an overview of existing deviations.
During our field research, we will also study what type of information auditors use to assess the risk of deviation types. This insight will be used in a follow-up phase to go from deviation to risk classification. Moreover, the correlation between risk level and the complexity of each deviating trace (i.e., the number and the variety of the mismatches in each deviating trace), in a real life setting will be investigated.

References

Capturing Resource Behaviour From Event Logs

Marijke Swennen\textsuperscript{1}, Niels Martin\textsuperscript{1}, Gert Janssenswillen\textsuperscript{1,2}, Mieke Jans\textsuperscript{1}, Benoît Depaire\textsuperscript{1}, An Caris\textsuperscript{1}, and Koen Vanhoof\textsuperscript{3}

Hasselt University, Agoralaan Bldg D, 3590 Diepenbeek, Belgium
Research Foundation Flanders (FWO), Egmontstraat 5, 1060 Brussels, Belgium
\{marijke.swennen, niels.martin, gert.janssenswillen, mieke.jans, benoit.depaire, an.caris, koen.vanhoof\}@uhasselt.be

Abstract. Process mining mainly focuses on the retrieval of process models from event logs. As these discovery algorithms make assumptions, performance analyses based on these models can present a biased view. In literature, algorithm-agnostic process metrics have been introduced. Given the critical importance of resources in the light of continuous process improvement, this paper extends the process metrics framework towards the resource perspective. New metrics are added and existing metrics are adapted. Using the extended framework, organisations can retrieve useful insights in resource behaviour.

Keywords: Process mining, Operational excellence, Resource behaviour, Log-based process metrics

1 Introduction

Process mining aims to retrieve valuable insights in business processes from event logs. Discovery, one of three types of process mining, aims to discover process models from event data. Related to control-flow, a plethora of algorithms have been developed \cite{1}, which all make particular assumptions and have a representational bias. Carrying out process performance analyses using such a process model entails the risk of presenting a biased view on the underlying process.

In this respect, the need for unbiased, algorithm-agnostic information retrieved from event logs is advocated in \cite{16}. Hence, a set of metrics, related to time and structuredness of the process, is identified. These unbiased insights in process behaviour can support performance measurement \cite{12} and continuous process improvement \cite{11} which is related to methodologies such as lean management and Six Sigma.

A classification of resources in the domain of project management can be found in \cite{7}. However, this paper focuses on process participants, software systems or equipment, as resources are defined in the field of BPM \cite{4}. The importance of resources is not yet recognised in \cite{16}. Nevertheless, resources are a source of process variability and their behaviour is essential in the light of continuous process improvement \cite{2}. Consequently, this dimension should be taken into account to convey a more comprehensive picture on process behaviour to organisations. This is consistent with the research recommendation in \cite{14} as it targets the resource perspective in process mining.

Given the need to include the resource perspective, this paper extends the work in \cite{16} to include resource-related process insights. To this end, new metrics are proposed and existing metrics are redefined or complemented with additional analysis levels.
Within the context of quantifying the resource perspective using event logs, metrics that mainly focus on the relationship between resources are proposed by [15]. While the latter specifies metrics with the purpose of mining organisational models, [5] and [13] focus on defining resource behaviour measures. The current paper complements this as well as the recently introduced resource availability metrics [9]. Besides the general contribution of providing algorithm-agnostic resource insights to organisations, the extension of [16] can also support organisations in performing knowledge management, for instance when creating a knowledge map [3], or project management with applications such as resource levelling or resource allocation [8].

![Extended overview of log-based process metrics](image)

**Fig. 1.** Extended overview of log-based process metrics [16].
2 Resource metrics

Next to three new metrics concerning the concept of resource behaviour, adaptations to existing metrics in [16] are presented. Levels of analysis currently presented are the log level, trace level, and activity level. New levels of analysis related to resources are the resource level, representing characteristics of the resources executing the activities, and the resource-activity level in which resource-activity combinations are considered as introduced in [10]. All metrics have been implemented as functions in the R-package edeaR, which is available on the Comprehensive R Archive Network [6]. An overview of all metrics presented in [16] extended with the new work presented in the paper at hand is provided in Figure 1.

2.1 Running example

The metrics presented in this paper will be illustrated by applying them to the running example introduced in Figure 2. The latter visualises an event log for five patients undergoing five different medical activities executed by six distinct staff members of a hospital. The duration of activities is presented by their length.

2.2 New resource metrics

**Resource frequency.** The frequency of resources executing activities in a business process can be insightful, e.g., during company restructuring. In Figure 2, a resource is associated to on average 5.167 activity executions, with a minimum of 1 (resource R6) and a maximum of 8 (resource R4), implying that R4 is probably more active and fundamental for this process. At trace level, this metric is less informative because, even though the sequence of activities is the same, the resources executing them can differ. Other analysis levels are activities, resources and resource-activity combinations. For the latter, both the resource perspective and activity perspective can be useful for the relative number. R4 executes a surgery four times, representing 50 % of the total number of executions by R4 and 57.14 % of the total number of surgeries.

**Resource involvement.** The involvement of resources in cases can be of interest to, e.g., decide how “indispensable” they are. R5, for instance, only helps three patients, while R3 is involved in the process of 100 % of the patients. On the resource-activity level, the involvement of specific resource-activity combinations in cases is provided.

![Fig. 2. Visualisation of running example event log.](image-url)
**Resource specialisation.** Finally, the specialisation level of resources in a company demonstrates which resources are performing certain activities more than others. This information can be used to tackle challenges such as team selection or brain drain, as presented in [3]. Next to the log level, the resource level shows the absolute and relative number of distinct activities that each resource executes. R1 only executes two types of activities, examinations and radiotherapy, while he executes these activities six times in total. At the level of distinct activities, we find that examinations are always executed by the same resource, while radiotherapy is executed by everyone.

2.3 Adaptations to existing metrics presented in [16]

Looking at the processing time per case on the resource level provides a company an overview of the amount of time each resource spends on a case and which resources spend more time on cases than others. R3 spends on average around five times more time per patient than R5. However, comparing different resources or activities can be more insightful at the resource-activity level.

Extra levels of analysis can also be added to the structuredness metrics regarding start and end activities in a process. Which resources execute the first or last activity per case can be of interest for a company. Probably this person is the contact person for the patient or customer or is responsible for all communication.

Moreover, resource information was not taken into account in the metrics regarding rework. The metrics presenting self-loops and repetitions should take into account if the activity is redone by the same resource or by another one. Therefore, these metrics are rewritten according to four concepts, presented in Figure 3. A repeat self-loop is found in case 5, where radiotherapy is executed twice immediately after each other by R4. Case 2 holds a redo self-loop, where R4 executes radiotherapy after R3 executed it. Case 2 also holds a redo repetition, i.e. surgery is executed by R3 and R4 (but not immediately after each other). Finally, a repeat repetition is present in case 4 where R3 executes chemotherapy twice with an execution of radiotherapy in between.

<table>
<thead>
<tr>
<th>Immediately following</th>
<th>Not immediately following</th>
</tr>
</thead>
<tbody>
<tr>
<td>Same Resource</td>
<td>Repeat self-loop, Redo self-loop</td>
</tr>
<tr>
<td>Different Resource</td>
<td>Repeat repetition, Redo repetition</td>
</tr>
</tbody>
</table>

Fig. 3. Dimensions of rework metrics.

3 Conclusions and Future Work

This paper presents an extension of [16] towards the resource perspective. New metrics are presented and existing metrics are adapted. This way, it anticipates on the importance of resources in the light of continuous process improvement and the need to gain algorithm-agnostic insights in their behaviour.

Future work involves applying the metrics to a real-life event log and formulating managerial recommendations based on these insights. Moreover, studying the evolution
of metric values over time and examining the combined evolution of metrics can provide valuable knowledge. For instance, a decrease in processing time at resource-activity level combined with an increase in rework over the same time period might not be desirable for organisations.

References

Research Plans and Demonstrations
Accelerating Process Mining using Relational Databases

Alifah Syamsiyah, Boudewijn F. van Dongen, Wil M.P. van der Aalst

Eindhoven University of Technology, Eindhoven, the Netherlands
A.Syamsiyah@tue.nl, B.F.v.Dongen@tue.nl, W.M.P.v.d.Aalst@tue.nl

Abstract. Given the abundance of event data, the challenge of process mining today is to enable process mining in the large. This research aims to address scalability problem in terms of memory use and time consumption. To this end, we use relational databases as the framework to both store event data and do process mining analysis. We conduct a pre-computation of intermediate structures during insertion time of the data. Finally, we implement the existing process mining algorithms to be compatible with relational database settings.

Keywords: process mining, big event data, relational database

This document contains the PhD research plan and is organized as follows. In Section 1 we define what will be accomplished by eliciting relevant research questions. In Section 2 we present the background knowledge. In Section 3 we explain the significance of the research contribution. Then in Section 4 we describe the method adopted in the research. Finally, in Section 5 we present what we have done so far.

1 Research Questions

This work is conducted to answer these following research questions:

– How to deal with tremendous event data in process mining?
– How to do process mining analysis with event data taken from databases?
– How to gain performance benefit from relational databases in terms of memory use and time consumption?

2 Background

Process mining is introduced as a research discipline that sits between machine learning and data mining on the one hand and process modeling and analysis on the other hand. It can be viewed as a means to bridge the gap between data science and process science. The goal of process mining is to turn event data into insights and actions in order to improve processes [12]. Given the
rapid development of event data, the challenge is to enable *process mining in the large*. This work will be focused on the use of relational databases as a storage of event data and as an engine to pre-compute process mining metrics.

There are some works related to the use of databases in process mining. XESame [15] is one of the tools to extract event data from databases. This work provides an interactive interface where users can select data from the database and then match it with XES elements. The downside of this work is the lack of direct access to the database since it is only considered as a storage place of data.

Another technique for extracting event data from databases was presented in [3]. This work uses two types of ontologies. The first is called domain ontology which gives a high level view of the data stored in the database. The second is called event ontology which contains the XES structure. Data in databases is extracted through these ontologies using a well-establish technology called Ontology-based Data Access [7]. Although this work is promising, the performance issues make it unsuitable for large databases.

RXES was introduced in [13] as the relational representation of the XES standard for event logs. The work presents the database schema as well as some experiments showing that RXES uses less memory compared to the standard approach. RXES puts the initial stone for direct access to the database, however, this research has no longer continued.

In addition to database approaches, some other techniques for handling big data in process mining have been proposed [2,8,9], two of them are decomposing event logs [1] and streaming process mining [5]. In decomposition, a large process mining problem is broken down into smaller problems focusing on a restricted set of activities. Process mining techniques are applied separately in each small problem and then they are combined to get an overall result. This approach deals with exponential complexity in the number of activities of most process mining algorithms [11]. In streaming process mining, process mining framework ProM is integrated with distributed computing environment Apache Hadoop. Hence we can analyze event data whose size exceeds the computers physical memory. Streaming process mining also provides online-fashioned process mining where the event data is freshly produced, i.e. it does not restrict to only process the historical data as in traditional process mining. However, neither decomposition nor streaming are directly applicable to existing process mining technique. Both approaches require some changes in the algorithms.

3 Significance

Relational database is one of the technologies used in big data computing. This research uses relational databases as the framework to enable process mining in the large. We argue that relational databases are the most suitable approach for process mining compared to other types of databases. The XES standard requires a relational representation between its elements, for example, an event must belong to a trace and a trace is part of a log. Therefore, aggregate-
oriented NoSQL databases \[14\] such as key-value store databases, document databases, and columned-oriented databases are not appropriate for XES event data. Relation-oriented NoSQL such as graph databases may be suitable, however, it does not provide supports for complex queries such as trigger.

Given the result from this research, process mining is able to handle big event data for discovering process models, doing conformance checking, and enhancing the process model. Moreover, process mining can be applied to the whole data to get insight from exceptional behavior.

4 Research Design and Methods

In this section we describe and motivate the method adopted in the research.

We first introduce a relational representation of XES, called DB-XES. Differently from normal process mining analysis which uses event log files, we use event data stored in relational databases. In other words, we move the location of data from files to databases. This provides scalability in terms of memory use due to the fact that memory is not bounded to the computer’s disk size.

Second, we move some computations from analysis-time to insertion-time. We pre-compute intermediate structures of process mining algorithms in the database and keep the computed tables up-to-date of the insertion of new events. Using this approach, we maintain the intermediate structure to be always ready.
and can be directly accessed by users whenever it is needed. This provides scalability in terms of time consumption since we cut the computation time inside process mining tools.

Figure 1 shows the DB-XES schema. As the XES structure [4], the schema contains main elements of event data, i.e. log, trace, event, and attribute. These elements are connected through table log_has_trace and trace_has_event. Global attributes, extensions, and classifiers are linked to the log. Furthermore, table event_collection is used to store the source of an event.

DB-XES also contains table dfr and log_has_dfr. This table is used to store Directly Follows Relation (DFR), i.e. a pair of event classes \((a,b)\) where \(a\) is directly followed by \(b\) in the context of a trace. DFR is one of the intermediate structures used by various process mining algorithms, such as Alpha Miner [10] and Inductive Miner [6].

For doing the experiment, we use real life event data from a company which contains 29,640 traces, 2,453,386 events, 54 different event classes, and 17,262,635 attributes. Then we extend this log in two dimensions, i.e. we increase (1) the number of event classes and (2) the number of traces, events and attributes. We extend the log by inserting copies of the original event log data with some modifications in the identifier, task name, and timestamp. We extend the number of event classes as a separate dimension since the growth of event classes gives exponential influences.

At the current stage, this work has limitation in the SQL query execution. The number of joins explodes and makes the query inefficient. Although the framework is still able to handle \(10^8\) number of traces, events, and attributes (the largest number used in the experiment), the need of optimizing the query still exists.

5 Research Stage

This research has been started since December 2015. In the first stage, we create a relational representation of XES called DB-XES. Then, using OpenXES as the interface, we create an access from DB-XES to ProM. Hence, any ProM plug-ins can work with DB-XES similarly as working with XES event log files.

In the next stage, we focus on enabling process discovery in large event data. We create a representation of the most common used intermediate structure, i.e. directly follows relations, in DB-XES. This structure is pre-computed and maintained to be up-to-date of the insertion of new events. Then, we conduct experiments using the state-of-the-art process discovery techniques, namely Inductive Miner. The result shows that the proposed solution gives performance benefit in terms of memory use and time consumption.

The experiment result is paving the way of applying other process mining techniques. In the current stage, we are implementing handover of work in DB-XES. The metrics have been translated into database tables, and some experiments are being run. In the following we briefly list the future research steps:
– Extend the approach with other advanced intermediate structures, such as the intermediate structures of declarative process mining.
– Apply the event removal feature in database while keeping the intermediate structures live under insertion and deletion of event data.
– Optimize the query performance through indexing and possibly apply more advanced big data technologies, such as Spark SQL.
– Implement conformance checking in the context of DB-XES.

References

Students’ Modeling based on their Problem Solving Behavior

Rabia Maqsood

Department of Computer Science, Università degli Studi di Milano, Italy
rabia.maqsood@unimi.it

Abstract. This research work aims at designing a framework to process the students’ logged traces and identifying different learning models based on their problem solving behavior specifically through trace-based exercises. Students depict different behaviors during problem solving including learning sequences and engagement level; thus yielding less structured and more complex interaction traces. It is therefore proposed to use Fuzzy Logic for pattern classification.

Keywords: Educational Process Mining, Pattern Classification, Problem Solving, Learning Traces.

1. Research Question

Teaching programming to freshers is a challenging job as most of the students fail to build conceptual models of complex programming concepts. The lack of ability to understand basic programming constructs at earlier stages results in a high failure or course drop out ratio in preliminary programming courses [8], [9]. Methods used to build viable mental models are commonly referred as: trace table, memory diagram and code visualization. These methods proved to be very effective as it helps novice programmers to better understand the flow of a program [9] and errors.

The core challenge is helping students in understanding the basic programming constructs and developing an accurate problem solving model, which is desirable by every teacher. Using trace tables for code dry-run exercises is amongst the traditional approaches used for teaching programming to the beginners [9]. Given a code snippet, students are required to fill a table showing each line of code to be executed in an expected manner and current state of the memory constructs (i.e. values of variables, arrays, etc.). However usually an instructor gets the final answer to the question(s) as in [13] and cannot keep track of the complete process followed by every individual. Consequently an instructor is not able to analyze the underlying process followed by each student [2].

Existing systems developed to support trace-based teaching keeps record of basic information only, for example: correctness/incorrectness of a solution and time spent on an exercise, as in [13] and [14]. However students usually perform several other operations during problem solving such as: adding/removing fields, changing values, submitting incomplete solution before the time expires, etc. Therefore analyzing the behavior of a student during problem solving can reveal the underlying process through interaction logs, for example: writing correct values of memory shows clarity of concepts, repeating a sequence of add/delete operations shows confusion or uncertainty about the answer, submitting incomplete solution earlier shows disengagement of a student during problem solving. Thus the problem solving process can vary to a large extent and students' behavior should be analyzed carefully to construct better students' models.

This research work aims at modeling students’ profile based on their problem solving behavior depicted through logged interactions in an e-learning tool. More specifically, this research study is intended to:

a. Use process mining (PM) techniques to analyze the learning process(es) followed by the students during solving trace-based exercises.

b. Classification of patterns discovered by process mining techniques.
c. And investigating the correlation of performance in trace-based exercises containing elementary concepts and exercises containing combination of concepts and thus yielding more complex concepts.

2. Background

Use of the technology to improve learning outcome has proved to be effective and is in use for almost more than two decades. With the immense use of technology in teaching, interactive e-learning tools are largely integrated into traditional classrooms [3]. Such an approach is referred to as ‘blended learning’ [6]. Intelligent e-learning tools are developed to support existing teaching practices as well as enhancing students’ learning by providing personalized learning environment to each student based on his/her learning profile. Thus creating accurate learner’s model is a critical issue in the development of intelligent e-learning tools and Intelligent Tutoring Systems [1]. A complete overview of existing approaches to create learner’s profile is given in [1] which includes: Item Response Theory (IRT), Bayesian Networks and its variants, Psychometric models and Knowledge space theory models.

Moreover researchers focused on Educational Data Mining (EDM) techniques to create better learners’ model. A growing interests in Educational Data Mining (EDM) has opened new challenges for the researchers to discover learning sequences [5] from interaction data recorded in e-learning tools (e.g. Intelligent Tutoring Systems (ITSs), MOOCs, etc.). Educational Data Mining (EDM) has been used extensively in the past to answer data-centric research questions such as: students’ retention, performance prediction, etc. [5], [6]. In general, EDM techniques deal with demographic data, academic history and current performance attributes only and thus are not able to identifying the underlying process(es) [6] followed by a student during problem solving. However it is evident from the work conducted in [5], [10], [12] that techniques developed for Process Mining (PM) seemed promising to discover learning sequences followed by extracting information hidden in event logs recorded for each user in the e-learning tools [6], [11].

Educational Process Mining (EPM) has evolved as a new research field recently, which focuses on extracting process-related information from event logs maintained in e-learning tools [4], [5], [6]. Computer based educational tools can record extensive information from student’s interaction with the system. The logged information in e-learning systems usually contains information about usage of learning material, assessment data [4], [7], system interaction history based on mouse-clicks, time spent on activities, etc. [10].

Prior research studies conducted in the context of EPM used different techniques to gain insights into the underlying learning process followed by the students during problem solving. For example, process mining techniques are used in [5] to identify learning sequences from students’ behavior logged during a scientific inquiry problem solving process and classify students based on their skills. Another study conducted in [6] discovered actual process followed by the students while solving Multiple-Choice Questions (MCQs). In [4], social mining techniques were used to analyze the interactions between training providers and courses involved in students’ training paths. Two groups of students were identified using Fuzzy miner in [12]: Surface and Deep learners, depicting different learning strategies followed by the students during writing a research project report. Authors in [10] also provide evidence of using logged traces to analyze different patterns adopted by the students during problem solving. A tree based algorithm is proposed in the reported work to predict with high accuracy a student’s likelihood of attempting problems of unfamiliar topics. In [11] a technique is presented that transforms sequences of logged events into more meaningful actions and activities, which are then classified into activities using heuristic miner. The results demonstrated the possibility of constructing more accurate students’ models from inquiry patterns using abstractions at various levels.

3. Significance

With the immense use of e-learning tools in traditional classrooms, it is becoming necessary to make better use of logged data to understand students’ behaviors. Students depict different behaviors
during problem solving including learning sequences (selection of problem sets), different order of performing tasks and engagement level; thus yielding less structured and more complex interaction traces. Prior work on investigating problem solving behavior involves writing skills [12], process inquiry skills [11], solving mathematical word problems [10], and very little or no attention is given on analyzing problem solving skills exercised by the programmers.

Earlier studies used process mining techniques to primarily deal with operational process(es) or structured data, considering students’ interaction logged data as input to the process mining algorithm [11], for example as in [4], [6] and [7]. However this is not the case when we have to investigate more complex human behaviors through their logged interactions. This data tends to be less structured [11], [12] as the order of performing task is not restricted and students may perform different intermediate actions before reaching to a final solution. Therefore it is proposed to use Fuzzy Logic for pattern classification which will be described in terms of linguistic variables and if-then rules.

4. Research design and methods

It is proposed to design a tool allowing students to solve trace-based programming exercises and automatically checking their solutions as described in [13]. Moreover the tool will maintain students' profiles which will be determined by processing the logged data. The tool will record data from students’ interaction including mouse-clicks, entries in the text box(es), adding/deleting rows in the solution trace table, etc. Students’ profile will be updated based on their performance and output class determined by the mining algorithm using their problem solving behavior.

5. Research stage

This is an initial proposal of the research study and will be conducted along the following phases:

a. Phase I: Initially a tool will be designed which will generate a solution trace table for each exercise by parsing the code into tokens and recording each new identifier and its attributes in a data structure, similar to how it is being done in a compiler. The tool will also maintain students’ profile by recording their interactions and performance attributes. In parallel a careful study of state-of-the-art techniques will be done and taking relevant courses in the first year.

b. Phase II: The tool will be used by the undergraduate students of Computer Science at the University of Milan, Italy, for experimental studies in several phases. The tool will maintain students’ profiles which will be updated based on their performance and output class determined by the mining algorithm using their problem solving behavior.

c. Phase III: Collected data will be studied carefully and appropriate process mining algorithm(s) will be used for pattern classification. Results will be analyzed and compared with existing approaches to model students’ profile.

References