3rd International Symposium on

Data-Driven Process Discovery and Analysis SIMPDA 2013

August 30, 2013
Riva del Garda, Italy

Editors:
Paolo Ceravolo
Rafael Accorsi
Philippe Cudre-Mauroux

Foreword

The third edition of the International Symposium on Data-driven Process Discovery and Analysis (SIMPDA 2013) conceived to offer a forum where researchers from different communities and the industry can share their insight in this hot new field.

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 - 2.12 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.

This year the symposium will be inserted among the VLDB 2013 workshops and will feature a number of presentations on recent research results and competitive PhD seminar. All this in the charming setting of Riva del Garda at the north-western corner of Lake Garda, at the southern edge of the Italian Alps, near the Dolomites.

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 third edition, 9 papers were submitted that were reviewed by a minimum of two reviewers; according to the format of a symposium the discussion during the event is considered a valuable element that can help to improve a work presented and the approach in presenting results. For this reason authors of accepted papers will be invited to submit extended articles a post-symposium volume of LNBIP (Lecture Notes in Business Information Processing), scheduled in 2014.

Our thanks go to the authors who submitted to the conference, to the board of reviewers that made a great work in the review process and in promoting this new event, and to all those who participate in the organization of the events.

We are very grateful to Università degli Studi di Milano, IFIP for their financial support, and to the University of Fribourg, the University of Freiburg, and ASSERT4SOA project.

Paolo Ceravolo
Rafael Accorsi
Philippe Cudre-Mauroux
SIMPDA co-Chairs

Table of Contents

* Research Papers

Sequential Approaches for Predicting Business Process Outcome and Process Failure Warning
Mai Le, Detlef Nauck, Bogdan Gabrys
pp. 1-15

Graph-Based Business Process Model Refactoring
María Fernández-Ropero, Ricardo Pérez-Castillo and Mario Piattini
pp. 16-30

Studies on the Discovery of Declarative Control Flows from Error-prone Data
Claudio Di Ciccio, Massimo Mecella
pp. 31-45

Development of a knowledge base for enabling non-expert users to apply data mining algorithms
Roberto Espinosa, Diego García-Saiz, Marta Zorrilla, Jose Jacobo Zubcoff, Jose-Norberto Mazón
pp. 46-61

Using Semantic Lifting for improving Process Mining: a Data Loss Prevention System case study
Antonia Azzini, Chiara Braghin, Ernesto Damiani, Francesco Zavatarelli
pp. 62-73

Challenges of Applying Adaptive Processes to Enable Variability in Sustainability Data Collection
Gregor Grambow, Nicolas Mundbrod, Vivian Steller, Manfred Reichert
pp. 74-88

Enhancing the Case Handling Paradigm to Support Object-aware Processes
Carolina Ming Chiao, Vera Künzle, Manfred Reichert
pp. 89-103

Knowledge and Business Intelligence Technologies in Cross-Enterprise Environments for Italian Advanced Mechanical Industry Project Presentation
Francesco Arigliano, Antonia Azzini, Chiara Braghin, Antonio Caforio, Paolo Ceravolo, Ernesto Damiani, Vincenzo Savarino, Claudia Vicari, Francesco Zavatarelli
pp. 104-110

On Process Rewriting for Business Process Security
Rafael Accorsi
pp. 111-126

Conference Organization

* Conference Co-Chairs

Paolo Ceravolo

Università degli Studi di Milano, Italy

Philippe Cudre-Mauroux

University of Fribourg, Switzerland

Rafael Accorsi

University of Freiburg, Germany

* Advisory Board

Karl Aberer, EPFL, Switzerland

Ernesto Damiani, Università degli Studi di Milano, Italy

Tharam Dillon, La Trobe University, Australia

Dragan Gasevic, Athabasca University, Canada

Marcello Leida, EBTIC (Etisalat BT Innovation Centre), UAE

Erich Neuhold, University of Vienna, Austria

Maurice van Keulen, University of Twente, The Netherlands

* PhD. Award Committee

Gregorio Piccoli, Zucchetti spa, Italy

Paolo Ceravolo, Università degli Studi di Milano, Italy

Marcello Leida, EBTIC (Etisalat BT Innovation Centre), UAE

* Web Chair

Fulvio Frati, Università degli Studi di Milano, Italy

Program Committee

Peter Spyns, Free University of Brussels, Belgium

Irene Vanderfeesten, Eindhoven University of Technology, The Netherlands

Daniele Bonetta, Università della Svizzera Italiana, Swizerland

Sylvain Hallé, Université du Québec à Chicoutimi, Canada

Ebrahim Bagheri, Ryerson University, Canada

Mustafa Jarrar, Birzeit University, Palestinian Territory

Valentina Emilia, Balas, University of Arad, Romania

Gregorio Martinez Perez, University of Murcia, Spain

Mohamed Mosbah, University of Bordeaux, France

Jan Mendling, Wirtschaftsuniversitat Wien , Austria

Maurice van Keulen , University of Twente,, The Netherlands

Gabriele, Ruffatti , Engineering Group, Italy

Eduardo Fernandez-Medina, University of Castilla-La Mancha, Spain

Chi Hung, Tsinghua University , China

Jose M. Alcaraz Calero, Hewlett-Packard Labs, United Kingdom

Richard Chbeir, University of Bourgogne, France

Mohamed Achemlal, Orange Labs, France

Helen Balinsky, Hewlett-Packard Laboratories, UK

Karima Boudaoud, Ecole Polytechnique de Nice Sophia Antipolis, France

Meiko Jensen, University, Germany

Renato Iannella, Semantic Identity, Australia

Farookh Hussain, University of Technology Sydney, Australia

Marcello Leida, EBTIC (Etisalat BT Innovation Centre), UAE

Frédéric Cuppens, Telecom Bretage, France

Gerd Groner, University of Koblenz, Germany

Abder Koukam, University of Technology, UTBM France

Wei-Chiang Hong, Oriental Institute of Technology, Taiwan (China)

Sponsors

Università degli Studi di Milano, Italy

University of Fribourg, Switzerland

University of Freiburg, Germany

Sequential Approaches for Predicting Business Process Outcome and Process Failure Warning

Mai Le[1], Detlef Nauck[2], and Bogdan Gabrys[1]

[1] Bournemouth University, Bournemouth, UK
mai.phuong@bt.com,bgabrys@bournemouth.ac.uk
[2] BT research, Ipswich, UK
detlef.nauck@bt.com

Abstract. Large service companies like telecommunication businesses run complex customer service processes in order to provide communication services to their customers. The flawless execution of these processes is essential since customer service is an important differentiator for these companies. They must also be able to predict if processes will complete successfully or run into exceptions in order to intervene at the right time, pre-empt problems and maintain customer service. Business process data is sequential in nature and can be very diverse. Thus, there is a need for an efficient sequential forecasting methodology that can cope with the diversity of the business data. In response to these requirements, in this paper we propose an approach which is a combination of KNN (K nearest neighbour) and sequence alignment for predicting process outcome. The proposed approach exploits temporal categorical features of the extracted data to predict the process outcomes using sequence alignment technique, and also addresses the diversity aspect of the data by considering subsets of similar process sequences, based on KNNs. We have shown, via a set of experiments, that our model offers better results when compared with original KNNs and random guess-based methods. We also introduce a rule based technique based on GOSPADE, which detects the repetitions of individual tasks and investigates the relationship between them and the process failure. The results are demonstrated in a comprehensive performance study on real business process data sets.

1 Introduction

Process mining is a relatively young research discipline but has been attracting growing attention recently due to its capability for predicting process outcomes. In every organisation, there are a large number of business processes to be dealt with on a daily basis. A business process is a sequence of tasks/activities that need to be completed to produce a product or to provide a service. Business processes need to be carefully monitored and controlled (business process management) to be effective and efficient.

The idea behind process mining is to discover, monitor and improve processes, aiming for excellent process execution. Process mining models contribute to many phases in business process management including the diagnosis phase,

operational support, etc. [1], [2], [3]: during the execution phase the process is monitored and can be slightly adjusted without redesigning the process; in the diagnosis phase, the enacted process is analysed and the outcome of this analysis can be used to redesign the process; Predictions and recommendations based on models learnt from historical data can be used for online maintenance because being aware of what might happen helps managers to set up suitable strategies to intervene on time [4]. For example, it is of interest to investigate certain loops that might lead to process failure; suggesting an optimal way to complete the process starting from the current step etc.

Such models can be drawn from a rich source of mathematical models in data mining [5]. However, due to the complexity and diversity of process data as well as the sequential nature of process data, it seems to be poorly aligned with any single classical data mining technique. Similar to data from business process executions, customer behaviour data also consists of asynchronous sequences and is considered as a kind of stochastic process. Customer behaviour data is described by sequences of interactions between the customers and the company while business processes are described by sequences of process events, i.e. steps or tasks. In both cases the events and the entities (customers or jobs) that move through the process instances are described by additional attributes. Therefore, in this study we use both types of data, business process data and customer behaviour data in order to test out if our proposed model is generic in terms of dealing with sequential data.

This paper introduces a new K nearest sequence method which is able to deal with sequential data. Our objective is to group similar sequences together expecting that sequences which behave similarly in earlier steps go to the same final step. In order to create a KNN model, which addresses the diversity and the temporal character of the data, an original KNN is combined with sequence alignment technique.

In our area of application we consider a number of sequences in which we want to find K sequences which are most similar to a given sequence $S(s_1^{(j)}, \ldots, s_{n_j}^{(j)})$, where j is the identity of the sequence. The similarity is determined using a distance function and, the prediction should be the majority class among classes of the K nearest sequences. This method is used in [6] to predict churn using data from a telecommunication company. The authors combined KNN and the theory of survival. In their work, Euclidean distance is used to calculate the distance between the given sequence and all sequences in the data sample. As our data consists of event sequences, in this study we use the edit distance in the process of comparing two sequences.

It is also of interest to look into loops, which occurred in the data, to identify some of them might lead to process failure. Hence, apart from the proposed predictive models, in this paper we employ a potential technique in order to provide warnings about process failure. It is based on a special algorithm in association rules called GOSPADE [7] and it enables us to detect loops and the links between them and the process failure.

The problem can be formulated as follows: A business process instance (S_j) is a composition of discrete events (or tasks) in a time-ordered sequence, $S_j = \left\{s_1^{(j)}, s_2^{(j)}, \ldots, s_{n_j}^{(j)}\right\}$, s_k takes values from a finite set of event types $E = \{e_1, \ldots, e_L\}$. Apart from its starting time $t_i^{(j)}$ and ending time $T_i^{(j)}$, each of these events has its own attributes. For simplicity, we assume that a process does not contain any overlapping events, that means there are no parallel structures.

The goal is to predict the outcome (success/failure) of a given process instance $S_{N+1} = \left\{s_1^{(N+1)}, s_2^{(N+1)}, \ldots, s_{i-1}^{(N+1)}\right\}$ based on the data from closed process instances $S = \{S_1, S_2, \ldots, S_N\}$ and to determine if the consecutive repetition of a task S_j potentially leads to a failure for the considered process instance.

The remainder of this paper is organised as follows: Section 2 presents the proposed models for predicting the process outcome. Section 3 discusses the GOSPADE algorithm based loop failure detection (LFD) technique. It is followed by Section 4 with a brief review of the data used in our analysis before we present the results of our experiments. In Section 5, we conclude the paper and discuss future directions.

2 Sequential KNNs

To determine sequence similarity both distance measure and similarity measure can be used. For numeric variables well-known distance measures exist and can be easily applied. However, in sequence analysis sometimes we have to work with sequences which are constructed from symbols, e.g. categories (churn prediction), phonemes (speech recognition) or characters (hand-writing recognition), etc. We need specialised functions which have the ability of measuring the similarity of symbolic sequences.

2.1 Sequence Alignment

Sequence alignment is very common in bio-informatics and has a relatively long history in this domain. The target entities of sequence alignment in bio-informatics are amino acid sequences of proteins, DNA sequences, etc. Sequence alignment is used for a number of purposes [8]. Algorithms used in sequence alignment are mainly divided into global alignment and local alignment. Global alignment provides a global optimisation solution, which span the entire length of all query sequences. In contrast, local alignment aims to find the most similar segments from two query sequences. In this work, both types of alignment are investigated to verify which one is effective in determining the similarity between the two sequences, the overall comparison between two given sequences or the most similar (consecutive) segments. The similarity between process sequences is used to predict the process outcomes.

– *Global algorithm*

In this kind of algorithms, sequences are aligned from the first event to the last one. One such algorithm was introduced by Needleman and Wunchs [8]. There are three characteristic matrices associated this algorithm: substitution matrix, score matrix and traceback matrix. The role of the substitution matrix is to generate the degree of matching any two events from the set of event types, or in other words matching subsequences of length 1. This degree which is irrespective of the position of the events then contributes to the matching score in the score matrix that consider the complete sequences, i.e. all events in the order they occur. Given two sequences, we then have to consider the order of the events and compute the score of matching the i^{th} event in one sequence with the j^{th} event in the other sequence, $i = \{1, \ldots, len_1\}$, $j = \{1, \ldots, len_2\}$ and len_1, len_2 are the lengths of the two given sequences. These scores define the score matrix. Finally, the trace back matrix encodes the optimal way of matching both sequences from a number of possible matches. We now introduce these three matrices.

1. Substitution matrix: in biology a substitution matrix describes the rate at which one amino acid in a sequence transforms to another amino acid over time. The entries of this matrix present the probabilities of transforming one amino acid to another. There are different ways of generating the substitution matrix. The simplest way is to not take into account the amino acid mutation factor, instead just give a score of 1 to the same amino acids and use a score of 0 to a pair of different amino acids.

$$s(i,j) = \begin{cases} 0 \; if & eventi \neq eventj \\ 1 & otherwise \end{cases}$$

   In this case, the substitution matrix is an identity matrix, the elements of the main diagonal are 1 and all the others are 0.
2. Score matrix: This matrix's elements are similarity degrees of events from the two given sequences.

$$h_{i0} = -\delta \times i, \tag{1}$$

$$h_{0j} = -\delta \times j, \tag{2}$$

$$h_{ij} = \max\left\{h_{i-1,j} - \delta, h_{i-1,j-1} + s(x_i, y_j), h_{i,j-1} - \delta\right\}, \tag{3}$$

   where $i = \{1, \ldots, len_1\}$, $j = \{1, \ldots, len_2\}$. δ is a specific deletion/insertion penalty value chosen by users. h_{i0} and h_{0j} are the initial values needed for the recursive formula in order to compute the entries of the score matrix h_{ij}. x_i and y_j are events at positions i and j from the given sequences. $s(x_i, y_j)$ is the score from the substitution matrix corresponding to events x_i and y_j.

3. Traceback matrix: Elements of this matrix are left, diag or up depending on the corresponding h_{ij} from the score matrix. These entries are built as follows.

$$q(i,j) = \begin{cases} diag\ if & h(i,j) = h(i-1,j-1) + s(i,j) \\ up & if \quad h(i,j) = h(i-1,j) - \delta \\ left & if \quad h(i,j) = h(i,j-1) - \delta \end{cases} \tag{4}$$

This matrix is used to track back from the bottom right corner to the top left corner to find the optimal matching path. Starting from the bottom right element, one moves in the direction given by the element which can be left, up or diag. This leads to another element with its own instruction (up, left or diag). By following the chain of directions the element at the top left corner is reached. The obtained path is the optimal way of matching the two given sequences.

– *Local algorithm*
The aim of local algorithms [9], [10] is to find a pair of most similar segments, from the given sequences. In this algorithm, a matrix of similarity degrees is built based on the following formula:

$$h_{i0} = h_{0j} = h_{00} = 0, \tag{5}$$

where h_{i0}, h_{0j} and h_{00} are the initial values for the recursive formula that is used to compute h_{ij}. Note that this is different from the global alignment. The initial values are set to be 0 because in local alignment it is not important where the common segment starts, the aim of the local alignment is to find the most similar segments of two given sequences.

$$h_{ij} = \max\left\{h_{i-1,j} - \delta, h_{i-1,j-1} + s(x_i, y_j), h_{i,j-1} - \delta, 0\right\}, \tag{6}$$

where $s(x_i, y_j)$ is the element of the substitution matrix as presented in the previous paragraph for global alignment.
The i^{th} event in a sequence can be aligned to the j^{th} event in another sequence, or can be aligned to nothing (deletion). This leads to a number of possible matchings of the two sequences. The optimal pair of aligned segments is identified by first finding the highest score in the matrix. This element is the end of the optimal aligned path. Then, the path is filled by tracking back from that optimal highest score diagonally up toward the left corner until 0 is reached.

2.2 KNNs Combined with Sequence Alignment

KNN is one of the classical approaches in data mining [11]. The model automatically selects sequences from the continuously updated data. KNN can be used as original non sequential approach [12] or extended into sequential approach [6]. The core idea is to find similar sequences expecting these sequences have a common behaviour and outcome. This expectation is understandable and is

common, for example in biology, similar DNA or protein sequences are hoped to have the same shape function.

Due to the sequential nature of business processes, we want to extract K similar sequences in terms of their temporal characteristics not numerical quantities. That is why the initial idea is to adopt the sequence alignment approach from biology and combine it with KNNs. To estimate the similarity between two sequences, the longest common segments between them can be used as criterion. The local sequence alignment can be applied directly. Alternatively, two sequences can be aligned globally, this finds the optimal matching by aligning from the first event to the last one of the two sequences. In this case, global matching allows us to compare the sequences in a whole rather than focus on their most similar consecutive segments. The resulting approach is named KNsSA (K nearest sequence with sequence alignment).

3 GOSPADE based Loop Failure Detection

Association rules are popular in areas like marketing, decision making and management support. They are used to detect information in the form of if-then rules or sequential patterns. For instance, retailers want to know which products customers usually purchase together (basket analysis). There are a number of studies in this area covering different aspects: rules of single level concepts, multiple level concepts, usefulness of rules and how to efficiently detect rules from a large data set with consecutive repeated tasks [13], [14], [11], [15], [7], [16], [17], [18].

3.1 Association Rule Mining

The aim of association rule mining is to find frequent patterns in a data set. Given e_i and e_j which are transactions or events, $e_i \Rightarrow e_j$ means if a sequence S_k in S contains e_i then e_j is contained in that sequence with a certain probability. In addition, the number of sequences in which $e_i \Rightarrow e_j$ appears is required to be larger than a given minimum support.

Definition 1. *The support of an event or association rule is the ratio of the number of sequences in the data sample which contain the considered event/rule with the number of all sequences.*

Definition 2. *The confidence of an association rule is the ratio of its support with the support of.*

There are a number of algorithms in association rules. Some of them were developed to capture the taxonomy structure of the problem, some of them are suitable to deal with repetition in the data:

- A-priori algorithm and some of its extensions [14] (basic).
- Generalizations of sequential patterns algorithm [18] (a-priori extension).
- ML (multiple levels) algorithm and some variations [15] (taxonomy).
- SPADE and GO-SPADE algorithms [7] (repetition).

3.2 GOSPADE Algorithm Variant for Loop Failure Detection

Intuitively, repetitions might slow down the completion of a process. Hence, we would like to look into the data and search for rules related to task repetition (loops) and the outcome of the process (success/failure). The GO-SPADE algorithm was designed to deal with consecutive data and it is suitable for detecting loops. We now introduce some concepts in association rule mining in order to help us to illustrate the principle of the GOSPADE algorithm which is briefly discussed afterwards:

Definition 3. *A rule of length k (composed of k elements) is called k-frequent sequence (pattern).*

Definition 4. *Prefix p of a a k-frequent rule z is the subsequence of z which consists of the first* $(k-1)$ *elements.*

Definition 5. *Suffix s of a rule z is its last element.*

Definition 6. *The location information of a pattern is listed and stored in a table, called idList. This information consists of the Id of the sequence in which the pattern occurred, the position of the pattern in the sequence and the rule itself (each pattern has its own idList). These sequence Ids and positions are denoted sid and eid respectively (please see Tables 1). While scanning through the data, each time the considered pattern occurs, a new row is added to the list for storing the information about the Id of the corresponding data sequence as well as the position of the pattern in the sequence. If the pattern is of length 1 then eid is the position of the task in the given sequence, otherwise eid is the position of the last task in k-frequent sequences when* $k > 1$*. To present the repetition of a pattern, instead of storing repeatedly it in different rows in the list, only one row is added to store the pattern by listing all of the eids. Since the information of the sequence Id and the pattern itself are the same for these repetitions. The eids of the replaced pattern are then written in interval form* $[i, i+j]$*, where* i*,* $i+1, \ldots, i+j$ *are the eids of the consecutive repeats of the pattern. If the pattern occurs only once at position* i*, we write* $[i, i]$*.*

The GO-SPADE algorithm consists of two stages:

- Generate candidates step: in the original algorithm, SPADE distinguishes two types of patterns, event patterns and sequence patterns. These types of patterns are determined based on the relationship between a prefix and its corresponding suffix. If the last item of the prefix p occurs at the same time as the suffix s, the pattern is called event pattern and is denoted ps. If the suffix s occurs strictly after the last item of the prefix p then the pattern is called sequence pattern and it is denoted $p \rightarrow s$. In our case, the available data consists of sequences of events. As mentioned in the introduction parallel structures are not considered and we are looking at sequence patterns. However, we work with loops only so the only IdList we have to investigate

is IdList of length 1. It then does not matter if we are dealing with event patterns or sequence patterns.
Let us consider event patterns for the sake of explaining the algorithm in a simple way. For more detail, refer to [7]. In the first step, candidate frequent patterns are generated. To generate frequent patterns of length $k+1$, all frequent patterns of length k which have the same prefix $k-1$ are considered. Once we have all k frequent patterns with the same $k-1$ prefix, the corresponding IdLists are used. Consider two k-sequences $z_1 = (p_1, s_1)$, $z_2 = (p_2, s_2)$, $p_1 = p_2$ and the corresponding IdLists IdList(z_1) and IdList(z_2). If prefixes p_1 and p_2 are the same, z_1 and z_2 are merged to generate a new frequent sequence z. For instance, the generated sequence is $z = p_1 s_1 s_2$.
To compute the IdList for the generated sequence IdList(z), a join operation is used. If sid1 is the same as sid2 then compare the eid of suffixes s_1, s_2 if eid1 is smaller than eid2 then chose eid2 as the eid of the new frequent sequence in case the support of generated sequence is bigger than minimum support.

– Counting support step: in GO-SPADE, it is easy to count the support of the generated sequences. It does not need to go through the data base to count the support of a candidate sequence. All the required information is already stored in its IdList.

Example: given four process instances (sequences) $P1$: $AAACB$ - success, $P2$: BAB - failure, $P3$: BCD - success, and $P4$: $ABBCDD$ - failure. Assume that we want to detect loops and just need to generate the IdLists of single events. The 1-IdList(A) generated from the given sequences is illustrated in Tables 1: Based on the IdLists for A, B, C and D, we select all the loops to generate a

sid	eid	Task	Loop
P1	[1, 3]	A	L
P2	[2, 2]	A	No
P4	[1, 1]	A	No

Table 1. The corresponding IdList of pattern of length 1 IdList(A).

new list as illustrated in Table 2. Based on the resulting list, we compute the percentages of failure and success. This GOSPADE based technique is the loop failure detection technique.

4 Evaluation

4.1 Data Preparation

We carried out a number of experiments based on records from three real processes ($DS1-3$) from a multi-national telecommunications company. In these datasets, the population of process instances turned out to be very diverse and

sid	eid	Outcome	Task	Loop	Support	Confidence
P1	[1, 3]	S	A	L	25%	1.0
P4	[2, 3]	F	B	L	25%	1.0
P4	[5, 6]	F	D	L	25%	1.0

Table 2. Loops and corresponding process outcomes found by LFD technique from a studied data set.

not straightforward to work with. $DS1$ represents a small scale set with only 10000 entries, 633 process instances, and hundreds of unique tasks. $DS2$ is also a real process with 11839 entries, 576 process instances with different lengths, and also has hundreds of unique tasks. The lengths of process instances in $DS2$ vary considerably. Last of all, $DS3$ is a customer behaviour data set used for churn prediction. This data set consists of 8080 customer records, each record is a sequence of events related to a customer. There are four types of events in churn data, churn, complaint, repair and provision. These customer event sequences are also of different lengths. Among these 8080 sequences, only 161 sequences contain a churn event. This shows the skewness of the data. Therefore, we artificially created new datasets with different ratios between non-churn and churn sequences in order to reduce the skewness of the data and to investigate if it had strong influence on the prediction model.

As we aim to predict the process outcome and to provide warning about process failure, it is necessary to label the outcome as either success or failure and this needs to be done before the proposed method can be applied. It is non-trivial to define process success or failure. In the case of $DS2$, the difference between the actual delivered date and the delivered date promised to the customer is used as the criterion to determine the success and failure. Particularly, if the actual delivered date is before the agreed date, that process instance is classified as success, otherwise, it is classified as failure. In contrast to $DS2$, $DS1$ and $DS3$ (churn and no churn labels) have available labels and therefore, they can be used directly as input for the proposed model.

4.2 Results for Predictive Models

To evaluate KnsSAs, we benchmarked our models with two other approaches:

- *RM - Random Model*: in order to find the outcome of the process, we randomly generate a number between 0 and 1, if the generated number is greater than 0.5 the outcome is success (1) and vice versa if the generated number is smaller than 0.5 the outcome is failure (0).
- *Original KNN*: we chose K nearest sequences in terms of having common unique tasks. For example, given two sequences ABD and AAC, there are one A, one B and one D in the first sequence, there are two A's and one C in the second one. Each unique task can be considered as one category, distance for each category is computed then the sum is taken to obtain the total distance between any two given sequences. For instance, the two

sequences given above consist of four categories A, B, C and D. The distance of category A is $d_A = 1$, and those of categories B, C, D are $d_B = 1$, $d_C = 1$ and $d_D = 1$ respectively. The resulting total distance of these two sequences is $d = d_A + d_B + d_C + d_D = 4$.

For the proposed models, we investigate the effect of K as it is important to get a reasonable number of similar sequences. As the labels are 0, 1, we decide to select odd values for K so we can always extract the outcome/label of the given sequence based on the K obtained sequences. The data sets $DS1$ and $DS2$ have a large number of unique tasks and the difference between the lengths of the sequences is substantial. Intuitively, the value of K should be small taking into account the diversity of the data. The results of the local KnsSA applying on data sets $DS1$ and $DS2$ are presented in Figure 1:

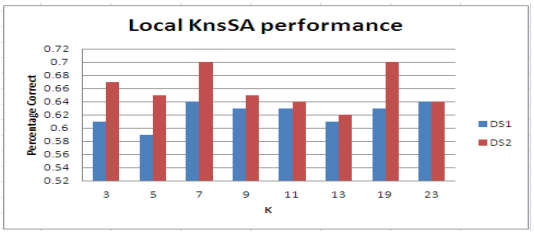


Fig. 1. Percentage of correct predictions of local KnsSA using data sets DS1, DS2.

We then tested our global KnsSA using the same datasets. The results are illustrated in Figure 2

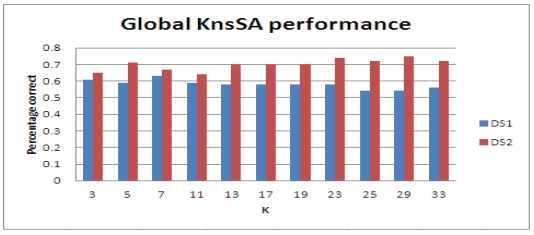


Fig. 2. Percentage of correct predictions of global KnsSA using data sets DS1, DS2.

Figures 1 and 2 show that both global KnsSA and local KnsSA are more accurate when they are applied to $DS2$. For $DS2$, global KnsSA with a higher value of K provides a better performance. When $K = 29$ the performance of the global KNsSA is 75%. On the other hand, applying global KnsSA to $DS1$ did not

achieve similar high performance. The highest correct percentage obtained is only 64% with $K = 7$. Moreover, when K is increased, global KnsSA's performance on $DS1$ decreases. This can be explained as $DS1$ is more diverse than $DS2$.

Global and local KnsSAs do not show any difference when they are applied to $DS1$. However, they show some difference in the case of $DS2$. This indicates that there are no important segments which have influence on the process outcome in $DS1$.

The performances of the original KNN, random guess and the proposed models, local and global KnsSAs, applied to the three data sets are presented in Figure 3:

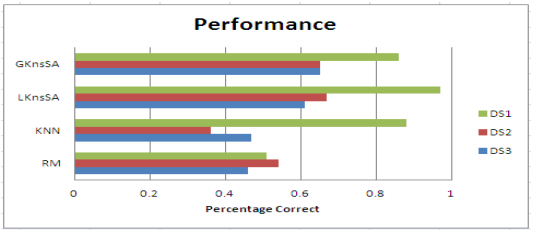


Fig. 3. Percentage of correct predictions of different models on data sets DS1, DS2 and DS3.

The results show that the proposed models outperform both benchmark models, original KNN and random guess. Especially, in the churn data, the proposed models capture churn event with a high degree of success whilst the other models do not. This also implies that the temporal characteristics of the data is important for predicting the process outcome.

We now present the results of the experiments by applying the two models, global and local KnsSAs to the churn data, $DS3$. The results are presented differently from the results for $DS1$ and $DS2$. This change is mainly made for the sake of dealing with the skewness of the churn data, the proportion between class 0 and class 1 in $DS3$ is 98:2 whilst that proportion for $DS1$ and $DS2$ is about 40:60. There are four tables which illustrate different aspects of the experiments' objectives. Table 3, shows the difference obtained by varying K, using local KnsSA and the original churn data. Table 4 demonstrates the results obtained by using local KNsSA when artificial data were created by changing the ratio between churn and no churn sequences in order to decrease the skewness of the data. Table 5 shows the performance of global KnsSA when applied to the former artificial data sets.

It can be seen from Table 3 that $K = 3$ is the best case for the churn data as our dataset is not big. Also, when the original data were modified in order to reduce its skewness, the performance of the model in terms of the churn detection objective improved even though the overall performance worsened. Intuitively,

Results/K	3	5	7
Total test data	809	809	809
Actual tests successful	793	793	793
Actual tests failure	16	16	16
Predicted tests successful	803	805	805
Predicted tests failure	6	4	4
Predicted tests success correct	793	793	793
Predicted tests failure correct	6	4	4
Correct ratio	0.99	0.99	0.99

Table 3. Local KnsSA applied on original DS3 with different K, $K = 3$, 5 and 7

Results/ratio	0.05	0.10	0.15
Total training data	511	875	1189
Total test data	59	100	135
Actual tests successful	45	83	120
Actual tests failure	14	17	15
Predicted tests successful	45	84	124
Predicted tests failure	14	16	11
Predicted tests success correct	44	80	118
Predicted tests failure correct	13	13	9
Correct ratio	0.97	0.93	0.94

Table 4. Local KnsSA applied on original DS3 with different churn and non churn ratios $K = 3$

when the population of churn sequences strongly dominates, it is very likely that our model could not catch the full churn set. It is shown in Table 3 that there are only 6 predicting cases with churn and all of them are correct. In the actual test data, there are 16 cases with churn. Although our model did not catch all of them, it achieves 100% correctness regarding the churn cases that it predicted. With the amended data, the overall performance of the model is reduced as well as the prediction rate for churn. Nonetheless, it is still of interest because out of 14 churn cases in the testing data, our model predicts 14 churn cases and 13 of them are correct. This amended data set consists of 161 (roughly 2% of the original data set) churn cases and 10% of the non churn cases of the original data set.

The results in Tables 4 and 5 show that the local KnsSA outperforms the global KnsSAs when applied to the churn data set. It might be caused by the fact that in customer behaviour sequences, only a set of special segments has strong influence on churn action.

4.3 Results for Warning Technique LFD

The results of the experiments on analysing the relationship between loops and process outcome using LFD applied to $DS1$, $DS2$ and $DS3$ are illustrated in the Tables 6, 7 and 8 correspondingly.

Results/ratio	0.05	0.10	0.15
Total training data	504	839	1208
Total test data	58	95	137
Actual tests successful	43	79	121
Actual tests failure	15	16	16
Predicted tests successful	48	90	134
Predicted tests failure	10	5	3
Predicted tests success correct	38	78	118
Predicted tests failure correct	5	4	0
Correct ratio	0.74	0.86	0.86

Table 5. Global KnsSA applied on original DS3 with different churn and non churn ratios with $K = 3$

Success-Loop 72.88%, Failure-Loop 27.12%						
sid	eid	Outcome	Task	Loop	Support	Confidence
9	[1, 2]	F	A	L	0.31	1.00
13	[2, 3]	S	A	L	0.17	1.00
17	[2, 3]	S	A	L	0.31	1.00
23	[2, 3]	S	A	L	0.31	1.00

Table 6. Loops and corresponding process outcomes found by the LFD in DS1.

Success-Loop 17.17%, Failure-Loop 82.83%						
sid	eid	Outcome	Task	Loop	Support	Confidence
416	[41, 42]	F	C	L	0.19	1.00
416	[1, 2]	F	B	L	0.21	1.00
421	[15, 16]	F	B	L	0.21	1.00
471	[16, 17]	S	B	L	0.21	1.00

Table 7. Loops and corresponding process outcomes found by the LFD in DS2.

Success-Loop 87.66%, Failure-Loop 12.34%						
sid	eid	Outcome	Task	Loop	Support	Confidence
1042	[1, 3]	F	4	L	0.89	1.00
1043	[1, 2]	S	4	L	0.89	1.00
1044	[1, 3]	S	4	L	0.89	1.00
1047	[1, 2]	S	4	L	0.89	1.00

Table 8. Loops and corresponding process outcomes found by the LFD in DS3.

As the goal is to verify if there is a link between a specified pattern and the outcome, specifically there is a link between a specified loop and failure, minimum support and minimum confidence are varied in order to filter out loops which are not frequent and not interesting. When minimum support is raised, only loops with high frequency are considered. In general, there is no loop in the data sets $DS1$ and $DS2$ which has high support and confidence.

The experiments don't show the link between loops and failure as expected whilst applying LFD to $DS1$. It is understandable as the loop task is 'contacting customer'. Obviously, people who executed this process tried to provide good service by repeating the task. In the case of $DS2$, the results of the experiments, in contrast, show the link between loops and failure. $DS3$ is a special case, there is no link between loops and failure or success because in general there are loops in each customer behaviour sequence, loops in task 2 and task 4 have 84% and 89% support respectively.

5 Conclusions

KNNs as a classical data mining approach have been widely used for modelling and predicting customer behaviour. This paper addresses some shortcomings of these predictive models, which occur when they are used for sequential data. Particularly, we propose some extensions to KNNs. These extensions were introduced in order to capture the temporal characteristic of the data and to profit from KNNs ability to deal with diverse data. Our extensions are tested on real business process data from a multi telecommunications company and the experiments provide some interesting results. First, the influence of global and local algorithms change depending on the data employed. For example, in the $DS1$ dataset, there is not much difference between using global KnsSA and local KnsSA. However, for $DS2$, global KnsSA is more accurate and in the case of $DS3$ local KnsSA outperforms global KnsSA. Even though the highest performance when predicting process outcome of the proposed models is just 75%, it outperforms the original KNNs, which proves that it is important to use sequential data in our problem.

Of all experiments, churn prediction shows the most interesting result. As churn is a rare event (2% of the data consist of churn), in the testing data set, there are only around 15 churn cases. Our models correctly capture most of these cases and percentage correct of churn prediction is very high. In the case of local KnsSA with $K = 3$, applied to $DS3$, out of 14 churn cases, the model predicts that there are 14 churn cases and 13 of them are correct.

It is interesting to see how the link between loops and process failure changes from case to case (different processes). In $DS1$, a specific loop implies that it is likely the process instance will end up with success. In $DS2$, there are certain loops which lead to process failure with high probability.

The experimental results encourage us to adopt the strategy of first grouping data into similar groups and then dealing with them using suitable approaches in our future work.

References

1. van der Aaslt, W.: Business process management: A personal view. Business Process Management Journal **10**(2) (2004) 135–139
2. van der Aaslt et al, W.: Process mining manifesto. In Daniel, F., Barkaoui, K., Dustdar, S., eds.: Business Process Management Workshops 2011. Volume 99 of Lecture Note in Business Information Processing. Springer, Berlin (2011)
3. Weijters, A., van der Aalst, W.: Process mining discovering workflow models from event-based data. In: ECAI Workshop on Knowledge Discovery and Spatial Data. (2001) 283–290
4. van der Aaslt, W., Schonenberg, M., Song, M.: Time prediction based on process mining. Information Systems **2** (2011) 450–475
5. Duda, R., Hart, P., Stork, D.: Pattern Classification. Wiley, New York (2001)
6. Ruta, D., Nauck, D., Azvine, B.: K nearest sequence method and its application to churn prediction. In Corchado, E., Yin, H., Botti, V., Fyfe, C., eds.: Intelligent Data Engineering and Automated Learning IDEAL 2006. Volume 4224 of Lecture Notes in Computer Science. Springer, Berlin (2006) 207–215
7. Leleu, M., Ligotti, C., Boulicaut, J., Euvrard, G.: Go-spade: Mining sequential patterns over datasets with consecutive repetitions. In: Conference on Machine Learning and Data Mining in Pattern Recognition. (2003) 293–306
8. Needleman, S., Wunsch, C.: A general method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of Molecular Biology **48** (1970) 443–453
9. Smith, T., Waterman, M.: Identification of common molecular subsequences. Journal of Molecular Biology **147** (1981) 195–197
10. Waterman, M.: Estimating statistical significance of sequence alignments. Philosophical Transactions of the Royal Society of London, Series B: Biological Sciences **344** (1994) 383–390
11. Berry, M., Linoff, G.: Data Mining Techniques: for Marketing, Sales, and Customer Relationship Management. Wiley, Newyork (2004)
12. Eastwood, M., Gabrys, B.: A non-sequential representation of sequential data for churn prediction. In: Proceedings of the KES2009 Conference, Santiago, Chile (2009)
13. Agrawal, R., Srikant, R.: Fasts algorithms for mining association rules in large databases. In: VLDB '94 Proceedings of the 20th International Conference on Very Large Data Bases. (1994) 487–499
14. Agrawal, R., Srikant, R.: Mining sequential patterns. In: ICDE '95 Proceedings of the Eleventh International Conference on Data Engineering. (1995) 3–14
15. Han, J., Fu, Y.: Discovery of multiple-level association rules from large databases. In: VLDB '95 Proceedings of the 21th International Conference on Very Large Data Bases. (1995) 420–431
16. Finding Interesting Rules from Large Sets of Discovered Association Rules. In: CIKM '94 Proceedings of the Third International Conference on Information and Knowledge Management. (1994)
17. Mohammed, J.Z.: Spade: An efficient algorithm for mining frequent sequences. Machine learning **42** (2001) 31–60
18. Srikant, R., Agrawal, R.: Mining sequential patterns: Generalizations and performance improvements. In: 5th International Conference on Extending Database Technology: Advances in Database Technology. (1996) 3–17

Graph-Based Business Process Model Refactoring

María Fernández-Ropero, Ricardo Pérez-Castillo and Mario Piattini

Instituto de Tecnologías y Sistemas de la Información, University of Castilla-La Mancha
Paseo de la Universidad 4, 13071
Ciudad Real, Spain
+34 926295300 Ext. 96697

[MariaS.Fernandez | Ricardo.PdelCastillo |
Mario.Piattini]@uclm.es

Abstract. Companies are ever more interested in process-oriented organizational designs due to the competitive advantages they can achieve. Companies must therefore be able to manage their business process models and deal with quality assurance concerns, especially when business process models are mined by reverse engineering (e.g. from information systems) since it has harmful effects on the quality. For example, non-relevant and fine-grained elements or incomplete processes can be obtained. Refactoring techniques have become a widely-used method to mitigate those effects, changing the internal structure of business process models without altering its external behavior. Business process models can be transformed into graph structures since it has been proved as a more efficient way to manage information. This work presents IBUPROFEN, a set of graph-based refactoring algorithms to improve the quality of business process models. This paper demonstrates its feasibility by conducting a case study using a set of industrial business process models.

Keywords: Business process model, refactoring, graph-based, understandability, modifiability, case study.

1 INTRODUCTION

Business processes depict the set of coordinated activities that companies have to conduct to achieve their common business goals [1]. Business processes are often represented by graph models following standard notations such as BPMN (Business Process Modeling and Notation) [2]. In these graph representations, business activities or tasks are considered as nodes and sequence flows between these tasks as edges. These standard representations provides companies with a mean to manage their business processes [3], i.e., analyze, execute and adapt their business process in an effective way. In fact, the appropriate management of business processes led to competitive advantages [4].

Unfortunately, business process models (as abstract descriptions) become misaligned regarding business processes that are daily, actually executed. This is due to daily operations change faster than business process models. In turn, this is owing to the fact that IT technologies and enterprise information systems evolve over time by adding new functionalities and operations that are not updated in business process models [5]. As a consequence, organizations are increasingly interested in quality assurance of business process representations and models they own.

There are several quality assurance techniques to achieve business process models with the appropriate quality levels. Business process mining techniques [6] are employed to obtain business process models from execution logs. Similarly, business process archeology [7] analyzes existing artifacts such as source code or databases for discovering and retrieving business process models in line with actual operation. Repairing techniques are devoted to add missing parts and correct business process models to fit them to the reality [8]. A part from all these techniques, one of the most applied and well-proven technique is business process model refactoring [9], which change the internal structure of business process models without altering or modifying their external behavior, and therefore, improving the understandability and modifiability among other quality features.

Despite standard notations such as BPMN are graph-based, most business process model refactoring techniques [10-12], hardly ever are designed as algorithms that manage graphs. Instead, most refactoring techniques consider, for example, business processes as two isolated, linear sets of business tasks and sequence flows. This design decision probably is better for the effectiveness of the refactoring algorithms but has harmful effects in terms of efficiency. This means that non-graph-based algorithms could have time-consuming problems when face with large, complex business process models. In fact, the usage of graph in different contexts [13-15] proved to be much more efficient than any other data structure. Due to this fact, this paper proposes IBUPROFEN, a business process refactoring approach based on graphs. IBUPROFEN defines a set of algorithms that are grouped into three categories according to the quality assurance challenge that address: maximization of relevant elements, reduction of fine-grained granularity and completeness. This paper depicts how business process models are managed as graphs and how are refactored according to the set of graph-based algorithms proposed in IBUPROFEN. This paper illustrates the usage of IBUPROFEN by means of a case study involving business process models obtained from real-life information systems, some of which are around 255 nodes and 512 edges.

The remainder of this paper is organized as follows: Section 2 summarizes related work; Section 3 introduces IBUPROFEN, the business process refactoring approach. Hence, their graph-based refactoring algorithms are shown as well as their implementation; Section 4 presents the application of the proposal with real-life business process models. Finally, Section 5 discusses conclusions and future work.

2 RELATED WORKS

There are various approaches in literature which deal with business process model refactoring by using different data structures. *Dijkman et al.* [10] identify refactoring opportunities in process model repositories. In order to identify similar parts in two different process models, these authors decompose both process models into smaller parts. They use a Refined Process Structure Tree (RPST) where the smaller parts of the decomposition are connected sub-graphs, such that control enters the fragment via a single entry node and leaves the fragment via a single exit node. The RPST defines a unique decomposition, i.e., a hierarchy of fragments that can be computed in linear time. The main difference of this work with our approach is that they use hierarchical structures and graphs in combination.

La Rosa et al. [12] provide a set of modularization of business process models based on graphs. This approach proposes, for example, horizontal modularization by obtaining sub-graphs of nearly equal size while keeping the number of cut edges low. However, not all the modularization and refactoring algorithms are based on graphs.

Similarly, the *Proviado* approach [16] applies a combination of graph reduction (Omission) and graph aggregation (Collapse) techniques to obtain customized process views based on a user query. The main difference of this work is that deal with visualization more than refactoring.

Weber et al. [11] enable designers to extract a process fragment into a sub-process to remove model redundancies, to foster reuse, and to reduce model size. Although this approach extracts fragments as sub-graphs, not all the refactoring algorithms proposed by these authors are based on graphs.

Finally, *Hauser et al.* [17] propose a process graph model to represent business process models as graphs and transform these graphs into executable code following the model-driven engineering principles. Unfortunately, the process graph model has not been used with refactoring purposes.

3 IBUPROFEN

IBUPROFEN (*Improvement and BUsiness Process Refactoring OF Embedded Noise*) addresses business process model refactoring. This technique has been especially designed for business process models represented according to the BPMN and mined by reverse engineering. IBUPROFEN allows applying different graph-based refactoring algorithms in order to address some of the challenges that involve this kind of business process models. For example, incompleteness is an important challenge to cope with since data can be distributed in several sources. Moreover, different types of granularity are a challenge to address because fine-grained granularity causes the quality degree is lower. Non-relevant information also causes a low degree quality since the model should not contain additional elements that do not perform any business logic in the organization.

With the aim to carry out the refactoring, business process models according BPMN are managed as graphs. Once business process models are represented as

graphs, a set of ten refactoring algorithms are performed. These ten refactoring algorithms supported by IBUPROFEN are divided into three categories regarding their purpose: maximization of relevant elements, fine-grained granularity reduction and completeness. For example, Fig. 1 shows one belonging to the first category, removing unnecessary elements. In that case, the gateway (that represents a decision node in BPMN) is removed since there are not nodes to choose, only one node (Task 2) can be executed after Task 1. Fig. 2 shows one belonging to the last category, completing the model. In that case, decision nodes (represented by gateways in BPMN) are added in incoming and outgoing branches. The diamond shape with the cross (exclusive gateway) represents that only one of the branches can be taken. The rest of refactoring algorithms can be consulted in [18].

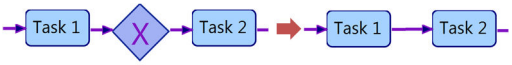


Fig. 1. Removing unnecessary nesting

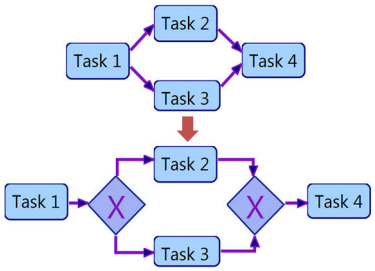


Fig. 2. Adding decision nodes between incoming and outgoing branches

IBUPROFEN is supported by a tool developed as an Eclipse™ plug-in (available in [19]). The supporting tool can therefore be used in combination with other Eclipse™ plug-ins aimed at obtaining business process models, e.g., from the source code of existing information systems. IBUPROFEN uses JGraphT [20] to manipulate graphs easily. JGraphT is a free Java graph library that provides mathematical graph-theory objects and algorithms. JGraphT supports various types of graphs such as, for example, directed graphs that are used in IBUPROFEN. Additionally, the BPMN file is read through the Jdom [21], library responsible for handling XML files via Java code.

The next sub-sections explain in detail the transformation between business process models according BPMN and graphs, as well as the implementation of each category of refactoring algorithms and the transformation from graphs to BPMN.

3.1 Transformation BPMN to Graphs

Each business process model is transformed into a directed weighted graph (`DefaultDirectedWeightedGraph<V,E>`), a non-simple directed graph in which

multiple edges between any two vertices are not permitted, but loops are. In our case, vertices (`V`) are modeled using the `BPElement` class while edges (`E`) are modeled using the `BPEdge` class. Both classes implement the `Cloneable` interface. `BPElement` saves the information about a BPMN element such as a task, data object, gateways and events. `BPEdge`, in turn, saves the information about an edge in the business process model such as a sequence flow and an association flow. The information that is saved by these classes is the name, the type, an identifier, among other. In case of events and gateways, nodes save the subtype of this type of node, i.e., start, intermediate and end for events and exclusive, parallel, inclusive and complex for gateways. Compounded tasks are, in turn, directed weighted graphs.

The set of business process models mined from existing information systems is transformed therefore to a list of graphs. Each graph represents one business process model and has several `BPElement` instances connected by means of several `BPEdge` instances. Thus, a business process model is represented as a graph through the notation G= (V, E), being v_1, $v_2 \epsilon$ V (nodes) and e ϵ E (edge), the edge between these nodes e = (v_1, v_2). For example, the connection between a task t1 and task t2 (sequence flow) is represented as follow: e = (v_1, v_2), e.type=SEQUENCE, e.name= "t1→t2", v_1.type=TASK, v_1.name=t1, v_2.type=TASK, v_2.name=t2.

3.2 Graph-based refactoring algorithms

IBUPROFEN provides a set of ten refactoring algorithms grouped into three categories: maximization of relevant elements, fine-grained granularity reduction and completeness [18]. The next paragraphs explain in detail the implementation of each refactoring algorithms belonging to each of the categories.

- **Maximization of relevant elements**.

This category groups the refactoring algorithms responsible for removing non-relevant elements found in business process models; these include isolated tasks, sheet tasks and inconsistencies. Moreover, nested gateways can bring about an increase in the complexity of business process models, so these are replaced by equivalent, light-weight structures. The refactoring algorithms are the following:

R1. Remove Isolated Nodes: This refactoring algorithm removes nodes (i.e., tasks, gateways or events) in the business process model that are not connected with any other node in that model, i.e., nodes without incoming and outgoing edges. Algorithm 1 illustrates this refactoring.

Algorithm 1: Remove Isolated Nodes.

Given G = (V, E)

$\forall$ a ϵ V

if $\neg\exists$ e ϵ E : (e = (b, a) $\vee$ e = (a, b), b ϵ V) then

V – {a}

R2. Remove Sheet Nodes: This refactoring algorithm removes elements in the business process model that are considered sheet nodes. These nodes can be gateways or intermediate events that have no successor nodes, i.e., nodes without outgoing edges in the graph. Algorithm 2 illustrates this refactoring.

Algorithm 2: Remove Sheet Nodes.

Given G = (V, E)

$\forall$ a ϵ V : a.type = INTERMEDIATE $\vee$ a.type = GATEWAY

if $\neg\exists$ e ϵ E : e = (a, b), b ϵ V then

V – {a}

R3. Merge nesting: This refactoring algorithm merges consecutive gateways of the same type (i.e., all gateways are exclusive or all are inclusive or parallel and so on). The merging is performed when the first gateway has only one output and the second has only one input. Algorithm 3 illustrates this refactoring algorithm.

Algorithm 3: Merge Nesting.

Given G = (V, E)

gatewaysToMerge $\leftarrow$ Ø

$\forall$ a ϵ V : a.type = GATEWAY

$\forall$ b ϵ V : □! e1 ϵ E : e_1 = (a, b)

if b.type = GATEWAY $\wedge$ □! e_2 ϵ E : v_2 = (b, a) $\wedge$ a.subType = b.subType then gatewaysToMerge $\leftarrow$ {(a,b)}

$\forall$ (g_1,g_2) ϵ gatewaysToMerge

$\forall$ e ϵ E : e = (g_2, a)

E $\leftarrow$ {e' = (g_1,a)}, E – {e}, V – {g_2}

R4. Remove Inconsistencies: This refactoring algorithm removes sequence flows in the business process model that are considered inconsistent. When two tasks are connected through a cut node, such as an intermediate event or a gateway, and through a direct sequence flow, this sequence flow is removed while the most restrictive path is maintained. Algorithm 4 illustrates this refactoring.

Algorithm 4: Remove Inconsistent Paths.

Given G = (V, E)

$\forall$ a ϵ V : a.type = GATEWAY $\vee$ a.type = INTERMEDIATE

$\forall$ e_s ϵ E : e_s = (a, a_s), a_s ϵ V

$\forall$ e_p ϵ E : e_p = (a_p, a), a_p ϵ V

if □ e ϵ E : e = (a_p, as) then

E – {e_s}, E – {e_p}, V – {a}

R5. Remove unnecessary nesting: This refactoring algorithm was shown in Fig. 1. It removes gateways that connect only two nodes, i.e. with one input and one output. This gateway is removed and a direct sequence flow is created between these nodes. Algorithm 5 illustrates this refactoring.

Algorithm 5: Remove unnecessary nesting.

Given G = (V, E)

$\forall$ a ϵ V : a.type = GATEWAY

if $\exists!e_s \epsilon E : e_s = (a, a_s) \wedge \exists!e_p \epsilon E : e_p = (a_p, a), a_s, a_p \epsilon V$ then

$E - \{e_s\}, E - \{ep\}, V - \{a\}, E \leftarrow \{e' = (a_p, a_s)\}$

- **Fine-grained granularity reduction**

The different granularity of business tasks and callable units in existing information systems constitutes another important challenge [22]. According to the approach proposed by [23], each callable unit in an information systems is seen as a candidate business task. However, existing systems typically contain thousands of callable units, some of which are large ones supporting the main business functionalities of the system, while many are very small and do not support any business activity directly. In other situations, a set of small callable units together support a business activity. This means that this category provides two refactoring algorithms to deal with large sets of fine-grained business tasks and data objects:

R6. Create compound tasks: This refactoring algorithm transforms each task into a compound task when this task has several subsequent tasks, which are in turn connected by a round-trip sequence flow to the task. This scenario comes about because each callable unit is transformed into a task during the reverse engineering stage when a given callable unit can invoke another callable unit, returning a value to the first one. In this case, a compound task is created with a start and end event connected with each subsequent task through the respective split and join exclusive gateways. Algorithm 6 illustrates this refactoring algorithm.

Algorithm 6: Create compound tasks.

Given G = (V, E)

$\forall$ a ϵ V : a.type = TASK

children $\leftarrow \emptyset$

$\forall$ b ϵ V : b.type = TASK

if $\exists$ $e_1 \epsilon E : e_1 = (a, b) \wedge \exists! e_2 \epsilon E : e_2 = (b, a)$ then

children $\leftarrow$ b

$V - \{b\}, E - \{ e_1\}, E - \{ e_2\},$

G' = (V', E')

V' $\leftarrow \{a_1, a_2, a_3, a_4\}$, a_1.type = START, a_2.type = END, a_3.type = COMPLEX, a_4.type = COMPLEX

E’ ← {m_1,m_2,m_3,m_4}, m_1.type = SEQUENCE, m_2.type = SEQUENCE, m_3.type = SEQUENCE, m_4.type = SEQUENCE, m_1 = (a_1,a_3), m_2 = (a_4, a_2)

∀ c ϵ children

E’← {m’,m’’}, m’.type = SEQUENCE, m’ = (a_3,c), m’’.type = SEQUENCE, m’’ = (c, a_4)

a.type = COMPOUND, a.subGraph = G’

R7. Combine data objects: This refactoring algorithm combines data objects that are input and/or output of a task. The combination is possible when those data objects are used exclusively (written or read) for that task. The combination is done when the number of data objects is above a threshold. To mitigate the collateral semantic loss, all the names of the grouped data objects are saved in the documentation attribute provided by the BPMN standard. Algorithm 7 illustrates this refactoring algorithm.

Algorithm 7: Combine data objects.

Given G = (V, E)

∀ a ϵ V : a.type = TASK

dataWrite ← Ø , dataRead ← Ø

∀ e ϵ E : e = (a, b), b ϵ V, b.type = DATA

if □! e ϵ E: e = (c, b) ∧ c=a then

dataWrite ← {b}

V – {b}, E – {e}

∀ e ϵ E : e = (b, a), b ϵ V, b.type = DATA

if □! e ϵ E: e = (b, c) ∧ c=a then

dataRead ← {b}

V – {b}, E – {e}

V ←{d_w}, d_w.type = DATA

∀ d_1 ϵ dataWrite

d_w.additionalInfo ← d_1.name

E ← {e_w}, e_w.type = SEQUENCE, e_w = (a, d_w)

V ←{d_r}, d_r.type = DATA

∀ d_2 ϵ dataRead

d_r.additionalInfo ←{d_2.name}

E ← {e_r = (d_r, a)}, e_r.type = SEQUENCE

- **Completeness**

Any reverse engineering technique implies an increase in the degree of abstraction, and therefore there is a semantic loss. This category is provided for that reason, to deal with semantic loss by means of the incorporation of additional elements that are not been retrieved in the reverse engineering phase. The refactoring algorithms are the following:

R8. Join Start and End events: This refactoring algorithm joins the start and end event to the starting and ending tasks, respectively. These events are created whether or not they were created before. When there are several starting tasks, the algorithm adds a split complex gateway between the start event and starting tasks. Similarly, if there are several ending tasks, it adds a joining complex gateway between ending tasks and the end event [24]. Algorithm 8 illustrates this refactoring.

Algorithm 8: Join start and end events.

Given G = (V, E)
∀ a ϵ V : a.type = TASK
    start ← Ø , end ← Ø
    if !□e ϵ E: e = (b, a), b ϵ V then
        start ←{a}
    if !□e ϵ E: e = (a, c), c ϵ V then
        end ←{a}
    V ← {v_1,v_2,v_3,v_4}, v_1.type = START, v_2.type = END, v_3 = COMPLEX, v_4 = COMPLEX
    E ←{e_1,e_2}, e1 = (v_1, v_3), e_2 = (v_4,v_2)
    ∀ v ϵ start
        E ←{e' = (v_3, v)}
    ∀ v ϵ end
        E ←{e' = (v, v_4)}

R9. Add gateways in incoming and outgoing branches: It is possible to obtain business process models by revere engineering that do not follow some of the good modeling practices that would be in accord with the BPMN standard as regards the usage of gateways [24]. In this case, this algorithm adds a split and join exclusive gateway when a certain task has several precursor or subsequent tasks, respectively (see Fig. 2). Algorithm 9 illustrates this refactoring.

Algorithm 9: Add gateways in branches.

Given G = (V, E)
∀ a ϵ V : a.type = TASK ∨ a.type = EVENT
    successor ← Ø, predecessor ← Ø
    ∀ b ϵ V :□e ϵ E: e = (a, b)
        if b.type = TASK ∨ b.type = EVENT ∨ b.type = GATEWAY
            successor ← {b}
            E – {e}
    ∀ b ϵ V :□e ϵ E: e = (b, a)
        if b.type = TASK ∨ b.type = EVENT ∨ b.type = GATEWAY
            predecessor ←{b}
            E – {e}

if |successor|>1 then

V ← {v_1}, v_1.type = EXCLUSIVE

∀ s ϵ successor

E ←{e_1,e_2}, e_1 = (a, v_1), e_2 = (v_1,s)

if |predecessor|>1 then

V ← {v_2}, v_2.type = EXCLUSIVE

∀ p ϵ predecessor

E ←{e_1,e_2}, e_1 = (p, v_1), e_2 = (v_1,a)

R10. Refine names: This refactoring algorithm implements a heuristic to improve labels of business tasks that were obtained almost directly from methods or functions of legacy source code through reverse engineering. These labels usually follow naming conventions present in most programming approaches such as the concatenation of various capitalized words. This refactoring algorithm split these labels into ones containing various words. This algorithm is not necessary to be shown due to the easy implementation using graphs.

3.3 Graph to BPMN

After applying refactoring algorithms, each graph is transformed into a business process model and each graph element is transformed into a BPMN element. Thus, each `BPElement` is transformed into a task, data object, gateway or event according to its type while each `BPEdge` is transformed into a sequence flow or association flow according to its type.

4 CASE STUDY

This section provides a case study with a real-life information system. The object of this case study is IBUPROFEN and the purpose of this case study is to evaluate how each refactoring algorithm affects to the understandability and modifiability of the business process model.

Despite the understandability depends on the people in charge of use, management or evaluation such business process models, i.e., it is subjective, understandability can be assess through several quality measures such as the number of nodes in the business process models, the number of nesting branches, the connectivity between elements, the density of elements, among others. For this reason, this paper considers as *dependent variables* the size, density and separability of a business process model in order to assess understandability and modifiability of a business process model.

- **Size** is the number of nodes in a business process model. This measure affects negatively to the understandability, i.e. a higher size difficult the understandability of a certain business process model [24].

- **Density** is the ratio between the total number of edges in a business process model and the theoretical maximum number of possible edges regarding the number of nodes. It affects the understandability and modifiability negatively, i.e., lower density values lead to business process models with a lower level of intricacy.
- **Separability** represents the ratio between the number of cut-vertices in a business process model (i.e. nodes that serve as bridges between otherwise strongly-connected components) and the total number of nodes. Separability affects the modifiability negatively, since higher separability implies hard and error-prone modifications of business process models.

The case under study is XCare information system. XCare is a mobile application of 9.9 thousands of lines of code. This application is intended for diabetes patients, which analyzes blood (through an external device) and suggests diet plans. Hence, the *independent variables* of this case study are each business process model obtained from XCare through reverse engineering.

The case study procedure consists of a set of semiautomatic steps that are executed in a computer with a 2.66 GHz dual processor and 4.0 GB RAM. The steps are as follows:

1. First of all, the extraction of business process model from XCare is performed by using MARBLE [25]. MARBLE is a tool used to recover business process models from existing Java code. This tool was selected because is released as an Eclipse plug-in and it therefore can be easily integrated with the IBUPROFEN tool.
2. Fig. 3 gives an example of a business process model obtained by MARBLE from XCare. This business process model contains 255 nodes and 512 edges, being the largest mined from XCare. The smallest model obtained has around 7 nodes and 6 edges. The sample can be visualized entirely and perfectly online [26]. Thus, a sample of 25 business process models is obtained from the source code from XCare.
3. The whole set of IBUPROFEN refactoring algorithms, that was mentioned in Section 3.2, are applied in each business process model retrieved in the above step. Refactoring algorithms are applied in isolation.
4. The dependent variables (size, density and separability) are recorded before and after applying each refactoring algorithm in order to be analyzed later.

Table 1 collects the value of the size, density and separability mean after applying each refactoring algorithm, as well as the gain obtained with respect to the original value. The gain is defined as the ratio between the difference of measure values and the original measure value. Thus, a positive gain means that the refactoring affects the measure positively while a negative gain means that the refactoring affects the measure negatively. A zero gain means that the value for a certain measure did not change after refactoring.

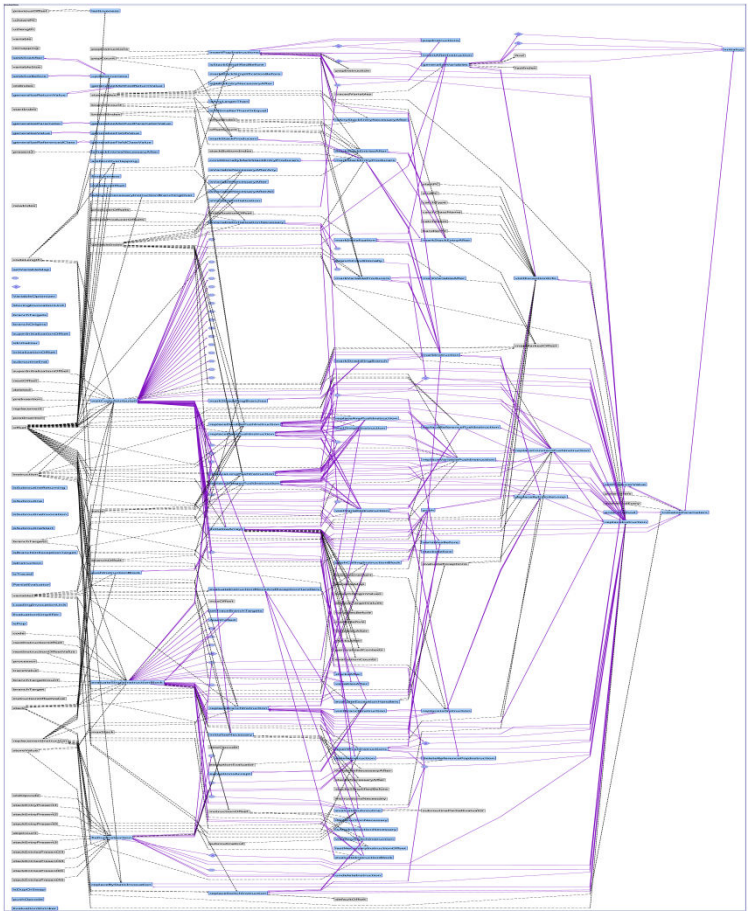


Fig. 3. Example of business process model managed by IBUPROFEN

Table 1. Effect of each refactoring algorithms on the size, density and separability

	Size		Density		Separability	
	Mean	Gain	Mean	Gain	Mean	Gain
Original	70.760	0.000	0.086	0.000	47.88	0.000
R1	46.440	0.366	0.150	-5.903	23.56	0.450
R2	67.120	0.030	0.087	-0.061	44.24	0.040
R3	70.760	0.000	0.086	0.000	47.88	0.000
R4	70.760	0.000	0.086	0.007	47.88	0.000
R5	70.440	0.003	0.086	-0.002	47.88	0.000
R6	62.560	0.114	0.067	0.093	48.76	-0.015
R7	63.040	0.098	0.093	-0.068	41.96	0.120
R8	74.360	-0.201	0.114	-0.511	51.48	-0.249
R9	90.160	-0.229	0.064	0.111	48.08	-0.002
R10	70.760	0.000	0.086	0.000	47.88	0.000

Table 1 reveals that removing isolated nodes decreases the size and separability while the density is increased. Despite the density is higher after R1, the relevance of the model has been increased since non-relevant elements have been removed. Similarly, R2 causes an increase of density when the size is decreased. Separability is decreased slightly. However, R3 has not impact on these measures due to business process models under study do not have nesting gateways. Removing inconsistencies (R4) maintains the same size and separability while the density is decreased because the number of edges is lower. Unnecessary gateways are removed (R5) and therefore the size is decreased while the density is increased owing to the number of nodes is lower. Separability after R5 is exacerbated slightly. R6 creates compound tasks in several business process models. This fact makes the size and density decrease in the most of cases. The same happens with R7, the number of nodes is lower but the number of edges is lower to and therefore, the density may increase while separability increases. R8 adds new missing elements in the model as start and end event as well as complex gateways. This makes that the size, separability and density are higher. In the same way, adding gateways in incoming and outgoing branches causes higher size. Nevertheless, the density after R9 tends to decrease due to there is more nodes in the model. Separability increases slightly since elements are more connected. In contrast, R10 does not have affect in any measures but the refinement of names implies an enhancement of the understandability.

5 CONCLUSIONS AND FUTURE WORK

Refactoring techniques has proved to be a good choice for improving business process models in terms, for example, of their understandability and modifiability levels. While graph-based algorithms have been successfully employed in different contexts, most business process model refactoring techniques often use alternative data structures [10-12] which leads to inefficient results. For this reason, this paper presents IBUPROFEN as a technique for refactoring business process models following a graph-based approach. Thus, the business process model is managed by means of a graph, changing its internal structure while its semantic is preserved. IBUPROFEN proposes ten refactoring algorithm divided into three groups in order to address the common problems that organizations have to deal when they retrieved such business process models by reverse engineering.

In order to demonstrate the feasibility of this approach, IBUPROFEN has been firstly implemented as an open source tool, and secondly, a case study with industrial business process models has been conducted. The case study reveals that the application the proposed graph-based refactoring algorithms improve the size, separability and density of business process models in the most of cases by removing non-relevant and fine-grained elements as well as by completing the models. The main limitation of this study is that results show size and density have an inverse relationship, i.e., when one increase the other decrease.

The second limitation lies in the empirical study analyzes the application of each refactoring algorithm in isolation. However, studies reveals that the order of applica-

tion of various refactoring algorithms in sequence could have an effect on the obtained results [18].

In line with the mentioned limitations, the future work will focus on the replication of the case study by analyzing alternative measures as well as the effect of different application orders. Furthermore, an algorithm improvement endeavor will be made to conciliate the size and density gain at the same time.

Acknowledgments

This work was supported by the FPU Spanish Program and the R&D projects MAGO /PEGASO (Ministerio de Ciencia e Innovación [TIN2009-13718-C02-01]) and GEODAS-BC (Ministerio de Economía y Competitividad & Fondos FEDER [TIN2012-37493-C03-01]).

6 REFERENCES

1. Weske, M., *Business Process Management: Concepts, Languages, Architectures*2007, Leipzig, Germany: Springer-Verlag Berlin Heidelberg. 368.
2. OMG. *Business Process Modeling Notation Specification 2.0*. 2011; Available from: http://www.omg.org/spec/BPMN/2.0/PDF/.
3. Jeston, J., J. Nelis, and T. Davenport, *Business Process Management: Practical Guidelines to Successful Implementations*. 2nd ed2008, NV, USA: Butterworth-Heinemann (Elsevier Ltd.). 469.
4. Davenport, T.H., *Need radical innovation and continuous improvement? Integrate process reengineering and TQM.* Strategy & Leadership Journal, 1993. **21**(3): p. 6-12.
5. Heuvel, W.-J.v.d., *Aligning Modern Business Processes and Legacy Systems: A Component-Based Perspective (Cooperative Information Systems)*2006: The MIT Press.
6. van der Aalst, W., *Process Mining: Overview and Opportunities.* ACM Transactions on Management Information Systems (TMIS), 2012. **3**(2): p. 7.
7. Pérez-Castillo, R., I. García-Rodríguez de Guzmán, and M. Piattini, *Business Process Archeology using MARBLE.* Information and Software Technology, 2011.
8. Fahland, D. and W.M.P.v.d. Aalst, *Repairing Process Models to Reflect Reality.* 2012.
9. Fernández-Ropero, M., R. Pérez-Castillo, and M. Piattini, *Refactoring Business Process Models: A Systematic Review*, in *7th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE)*, J. Filipe and L. Maciaszek, Editors. 2012, SciTePress: Wrocław, Poland. p. 140-145.
10. Dijkman, R., et al., *Identifying refactoring opportunities in process model repositories.* Inf. Softw. Technol., 2011. **53**(9): p. 937-948.
11. Weber, B., et al., *Refactoring large process model repositories.* Computers in Industry, 2011. **62**(5): p. 467-486.
12. La Rosa, M., et al., *Managing process model complexity via abstract syntax modifications.* Industrial Informatics, IEEE Transactions on, 2011. **7**(4): p. 614-629.
13. Pham, M.-D., P. Boncz, and O. Erling, *S3G2: A Scalable Structure-Correlated Social Graph Generator*, in *Selected Topics in Performance Evaluation and Benchmarking*, R. Nambiar and M. Poess, Editors. 2013, Springer Berlin Heidelberg. p. 156-172.

14. Gubichev, A. and T. Neumann, *Fast approximation of steiner trees in large graphs*, in *Proceedings of the 21st ACM international conference on Information and knowledge management*2012, ACM: Maui, Hawaii, USA. p. 1497-1501.
15. Mens, T., G. Taentzer, and O. Runge, *Analysing refactoring dependencies using graph transformation*. Software & Systems Modeling, 2007. **6**(3): p. 269-285.
16. Bobrik, R., M. Reichert, and T. Bauer, *View-Based Process Visualization*, in *Business Process Management*, G. Alonso, P. Dadam, and M. Rosemann, Editors. 2007, Springer Berlin Heidelberg. p. 88-95.
17. Hauser, R. and J. Koehler, *Compiling Process Graphs into Executable Code*, in *Generative Programming and Component Engineering*, G. Karsai and E. Visser, Editors. 2004, Springer Berlin Heidelberg. p. 317-336.
18. Fernández-Ropero, M., et al., *Assessing the Best-Order for Business Process Model Refactoring*, in *28th Symposium On Applied Computing (SAC)*2013: Coimbra, Portugal. p. 1400-1406.
19. Fernández-Ropero, M., R. Pérez-Castillo, and M. Piattini. *IBUPROFEN*. 2012; Available from: http://marketplace.eclipse.org/content/IBUPROFEN.
20. Naveh, B. and Contributors. *JGraphT*. 2011; Available from: http://jgrapht.org/.
21. Hunter, J., *JDOM*, 2009.
22. Pérez-Castillo, R., et al., *Generating Event Logs from Non-Process-Aware Systems Enabling Business Process Mining*. Enterprise Information System Journal, 2011. **5**(3): p. 301–335.
23. Zou, Y. and M. Hung, *An Approach for Extracting Workflows from E-Commerce Applications*, in *Proceedings of the Fourteenth International Conference on Program Comprehension*2006, IEEE Computer Society. p. 127-136.
24. Mendling, J., H.A. Reijers, and W.M.P. van der Aalst, *Seven process modeling guidelines (7PMG)*. Information and Software Technology, 2010. **52**(2): p. 127-136.
25. Pérez-Castillo, R., et al., *MARBLE. A Business Process Archeology Tool*, in *27th IEEE International Conference on Software Maintenance (ICSM'11)*2011, IEEE Computer Society: Williamsburg, Virginia, USA. p. 578-581.
26. Fernández-Ropero, M., R. Pérez-Castillo, and M. Piattini. *Extra material of Graph-Based Business Process Model Refactoring* 2013; Available from: http://alarcos.esi.uclm.es/per/mfernandez/material3.html.

Studies on the Discovery of Declarative Control Flows from Error-prone Data

Claudio Di Ciccio[12‡] and Massimo Mecella[1†]

[1] Sapienza – Università di Roma, Rome, Italy
[†] mecella@dis.uniroma1.it
[2] Wirtschaftsuniversität Wien, Vienna, Austria
[‡] claudio.di.ciccio@wu.ac.at

Abstract. The declarative modeling of workflows has been introduced to cope with flexibility in processes. Its rationale is based on the idea of stating some basic rules (named constraints), tying the execution of some activities to the enabling, requiring or disabling of other activities. What is not explicitly prohibited by such constraints is implicitly considered legal, w.r.t. the specification of the process. Declarative models for workflows are based on a taxonomy of constraint templates. Constraints are thus instances of constraint templates, applied to specific activities. Many algorithms for the automated discovery of declarative workflows associate to each constraint a support. The support is a statistical measure assessing to what extent a constraint was respected during the enactment(s) of the process. In current state-of-the-art literature, constraints having a support below a user-defined threshold are considered not valid for the process. Thresholds are useful for filtering out guesses based on possible misleading events, reported in logs either because of errors in the execution, unlikely process deviations, or wrong recordings in logs. The latter circumstance can be considered extremely relevant when logs are not written down directly by machines reporting their work, but extracted from other source of information. Here, we present an insight on the actual capacity of filtering constraints by modifying the threshold for support, on the basis of real data. Then, taking a cue from the results performed on such analysis, we consider the trend of support when controlled errors are injected into the log, w.r.t. individual constraint templates. Through these tests, we demonstrate by experiment that each constraint template reveal to be less or more robust to different kinds of error, according to its nature.

Keywords: process mining, artful process, declarative workflow, noisy event log

1 Introduction

Processes are typically represented as graphs, delineating their possible executions altogether, from the beginning up to the end. Most of the used notations are indeed derived by Petri Nets [2], such as Workflow Nets [1], BPMN [8], YAWL

[4]. The classical approach is called "imperative" because it explicitly represents every step allowed by the process model at hand. This leads to the likely increase of graphical objects as the process allows more alternative executions. The size of the model, though, has undesirable effects on the understandability and also on the likelihood of errors (see [18] for an insight of the Seven Process Modeling Guidelines): larger models tend to be more difficult to understand [19], not to mention the higher error probability which they suffer from, with respect to small models [17].

The declarative workflow models [22] have been introduced to cope with flexibility in processes. Its rationale is based on the idea of stating some basic rules (named constraints), tying the execution of some activities to either the enabling, requiring or disabling of other activities. What is not explicitly prohibited by such constraints is implicitly considered legal, w.r.t. the specification of the process. Declarative models for workflows are based on a taxonomy of constraint templates. Constraints are thus instances of constraint templates, applied to specific activities. A collection of constraints constitute altogether a declarative workflow. ConDec [20], now renamed Declare, is the most used language for modeling declarative workflows in the community of Business Process Management. It provides an extendible list of constraint templates, which we will consider in the remainder of this paper. Declarative models are particularly effective with some non-conventional kinds of process. For instance, professors, researchers, information engineers and all those professionals contributing to the production of a valuable but intangible products, such as knowledge, are commonly defined "knowledge workers" [23]. They are used to dealing with rapid decisions among multiple choices, based on their expertise, competence and intuition. There is an art in the management of their work. This is the reason for the name assigned to their processes: artful processes [14], which belong to the larger category of knowledge-intensive processes [9]. Artful processes are thus very flexible, dynamic and subject to change. Due to their characteristics, the declarative approach suits to their modeling [5]. Mining their workflow would be of extreme interest for understanding the best practices and winning strategies adopted by expert knowledge workers.

Process Mining [3], a.k.a. *Workflow Mining* [2], is the set of techniques that allow the extraction of process descriptions, stemming from a set of recorded real executions. Such executions are intended to be stored in so called *event log*s, i.e., textual representations of a temporarily ordered linear sequence of tasks. Many techniques have been proposed for mining Declare workflows ([16,15,10,11,7,6]). Most of them associate to each discovered constraint a support, i.e., a statistical measure assessing to what extent a constraint was respected during the enactment(s) of the process. Those discovered constraints having a support below a user-defined threshold are considered not valid for the process. Thresholds are useful for filtering out guesses based on possible misleading events, reported in event logs either because of errors in the execution, or due to very unlikely process deviations, or caused by wrong registrations of events in logs. The latter circumstance can be considered extremely relevant when event logs are not written down directly by machines reporting their work, but extracted from other sources of information. Artful processes, e.g., are known to be scarcely

automated [9]. Therefore, there are few possibilities to rely on classical system logs, keeping track of their executions. As a matter of fact, despite the advent of structured case management tools, many enterprise processes are still "run" over email messages. Artful processes, for instance, often require the collaboration of many actors, who usually share their information by means of email messages. Thus, email messages are a valuable source of information and event logs can be extracted out of them, relying on their content and meta-data (e.g., the delivery timestamp). [12] presents a novel approach and a tool, named MailOfMine, designed to mine declarative workflows for artful processes out of email collections. First, MailOfMine inspects subjects, bodies and headers of given archives of email messages: assuming that reading about the execution of an activity can be interpreted as the reporting of its actual enactment, it searches the email messages where one among a list of user-defined expressions is found. Each is considered an event. Then, considering the temporal ordering of email messages in every archive, a trace in the log is built accordingly. Such log is passed to the MailOfMine control flow discovery algorithm (MINERful), which returns the declarative model for the artful process laying behind the email communications analyzed. Extracting logs out of email messages leads to possible errors though, due to the automated interpretation of semi-structured texts. Hence, such extracted logs are intrinsically prone to errors. Thereby, mistakes in the discovered workflow are likely to increase.

This is actually the question we search an answer for in this paper: what happens to unknown models when they are discovered on the basis of logs which are affected by errors. [13] investigates an approach for repairing *process models* basing on event data. Conversely, we consider the possible unreliability of data which process models are discovered from, supposing that process models were not previously known at all. In this paper, we first report the analysis of the results obtained by applying MailOfMine to real data, focused on the precision of the inferred model with respect to the support threshold. Then, we present an insight on the trend of the support in presence of errors, injected into synthetic logs. We focus on different types of errors (insertion or deletion of events) and spreading policies (a given percentage per each trace or all over the log). We repeat our experiments for each of the possible constraint templates that the MINERful algorithm is able to discover. Thus, we aim at understanding the different levels of robustness that constraint templates show w.r.t. the different types of errors.

The remainder of the paper is as follows. Section 2 describes the constraint templates of Declare and their usage for describing a declarative process model. Section 3 reports the results of tests on real data (Section 3.1) and experiments conducted on the basis of tunable injection of errors into synthetic logs (Section 3.2). Section 4 concludes this paper and outlines the future paths for our investigation that this paper sheds light on.

2 The declarative process model

Here we abstract activities as symbols (e.g., ρ, σ) of an alphabet Σ, appearing in finite strings, which, in turn, represent process traces. We will interchange-

ably use the terms "activity", "character" and "symbol", as well as "trace" and "string", then. We adopt the subset of Declare taxonomy of constraints for modeling processes, as in [16]. For a comprehensive analysis of all the constraint templates in Declare, the reader can refer to [20,21].

Constraints are temporal rules constraining the execution of activities. E.g., $Response(\rho, \sigma)$ is a constraint on the activities ρ and σ, forcing σ to be executed if the ρ activity was completed before. Such rules are meant to adhere to specific constraint *templates*. *RespondedExistence* is the template of $RespondedExistence(\rho, \sigma)$. We further categorize constraint templates into *constraint types*. For instance, *RespondedExistence* belongs to the *RelationConstraint* type. Figure 1 depicts the subsumption hierarchy of Declare constraints.

Declare constraints are always referred to an activity at least, which we call "implying": if it is executed, the constraint is triggered – vice-versa, if it does not appear in the trace, the constraint has no effect on the trace itself. The $Existence(M, \rho)$ constraint imposes ρ to appear at least M times in the trace. We rename $Existence(1, \rho)$ as $Participation(\rho)$. The $Absence(N, \rho)$ constraint holds if ρ occurs at most $N - 1$ times in the trace. We call $Absence(2, \rho)$ as $Uniqueness(\rho)$. $Init(\rho)$ makes each trace start with ρ.

The aforementioned constraints fall under the type of *ExistenceConstraint*s, as they relate to an "implying" activity only. The following are named *RelationConstraint*s, since the execution of the implying imposes some conditions on another activity, namely the "implied".

$RespondedExistence(\rho, \sigma)$ holds if, whenever ρ is read, σ was either already read or going to occur (i.e., no matter if before or afterwards). Instead, $Response(\rho, \sigma)$ enforces it by requiring a σ to appear after ρ, if ρ was read. $Precedence(\rho, \sigma)$ forces σ to occur after ρ as well, but the condition to be verified is that σ was read - namely, you can not have any σ if you did not read a ρ before. $AlternateResponse(\rho, \sigma)$ and $AlternatePrecedence(\rho, \sigma)$ strengthen respectively $Response(\rho, \sigma)$ and $Precedence(\rho, \sigma)$ by stating that *each* ρ (σ) must be followed (preceded) by at least one occurrence of σ (ρ). The "alternation" is in that you can not have two ρ's (σ's) in a row before σ (after ρ). $ChainResponse(\rho, \sigma)$ and $ChainPrecedence(\rho, \sigma)$, in turn, specialize $AlternateResponse(\rho, \sigma)$ and $AlternatePrecedence(\rho, \sigma)$, both declaring that no other symbol can occur between ρ and σ. The difference between the two is in that the former is verified for each occurrence of ρ, the latter for each occurrence of σ. The reader should note that the hierarchy under the *Precedence* constraint template does not inherit the base and implied symbols from the *RespondedExistence* parent; it overrides them both by inverting the two, instead. This is due to the semantics of the constraints themselves.

The *MutualRelation* constraints follow: they are verified iff two *RespondedExistence* (or descendant) constraints (resp., (*forward* and *backward*, in Figure 1) are satisfied. $CoExistence(\rho, \sigma)$ holds if both $RespondedExistence(\rho, \sigma)$ and $RespondedExistence(\sigma, \rho)$ hold. $Succession(\rho, \sigma)$ is valid if $Response(\rho, \sigma)$ and $Precedence(\rho, \sigma)$ are verified. The same holds with $AlternateSuccession(\rho, \sigma)$, equivalent to the conjunction of $AlternateResponse(\rho, \sigma)$ and $AlternatePrecedence(\rho, \sigma)$,

and *ChainSuccession*(ρ, σ), with respect to *ChainResponse*(ρ, σ) and *ChainPrecedence*(ρ, σ).

Finally, we consider *NegativeRelation* constraints: they are satisfied iff the related *MutualRelation*s (*negated*, in Figure 1) are not. *NotChainSuccession*(ρ, σ) expresses the impossibility for σ to occur immediately after ρ (the opposite of *ChainSuccession*(ρ, σ)). *NotSuccession*(ρ, σ) generalizes the previous by imposing that, if ρ is read, no other σ can be read until the end of the trace (*Succession*(ρ, σ) is the *negated* constraint). *NotCoExistence*(ρ, σ) is even more restrictive: if ρ appears, not any σ can be in the same trace (the contrary of *CoExistence*(ρ, σ)).

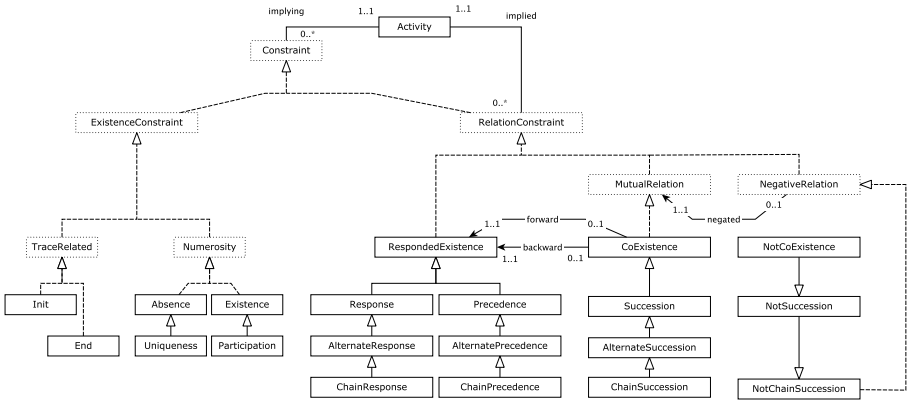


Fig. 1: The declarative process model's hierarchy of constraints. Taking into account the UML Class Diagram graphical notations, the Generalization ("is-a") relationship represents the subsumption between constraint templates. The subsumed is on the tail, the subsuming on the head. The Realization relationships indicate that the constraint template (and the subsumed in the hierarchy) belong to a specific type. Constraint templates are drawn as solid boxes, whereas the constraint types' boxes are dashed.

As a brief example, we may want to model the process of defining an agenda for a research project meeting. The schedule is discussed by email among the participants. We suppose that a final agenda will be committed ("confirm" – n) after that requests for a new proposal ("request" – r), proposals themselves ("propose" – p) and comments ("comment" – c) have been circulated.

The aforementioned activities are bound to the following constraints, then. If a request is sent, then a proposal is expected to be prepared afterwards (cf. *Response*(r, p)). Comments can be given in order to review a proposed agenda, or for soliciting the formulation of a new proposal. Thus, the presence of c in the trace is constrained to the presence of p (cf. *RespondedExistence*(c, p)). A confirmation is supposed to be mandatorily given after the proposal, and vice-versa any proposal is expected to precede a confirmation (cf. *Succession*(p, n)). We

suppose the confirmation to be the *final* activity (cf. *End*(n)). This mandatory task (cf. *Participation*(n)) is not expected to be executed more than once (cf. *Uniqueness*(n)).

Hence, the example process consists in the six aforementioned constraints: *Response*(r, p), *RespondedExistence*(c, p), *Succession*(p, n), *Participation*(n), *Uniqueness*(n) and *End*(n). As an example, the following traces would be compliant to the workflow: pn, pcn, rpcn, rpcpn, rrpcrpcrcpcn, rpprpcccrpcn.

3 Experiments and evaluation

In order to inspect the quality of the control flow discovery in presence of error-prone logs, we first verified the whole MailOfMine system on real data (Section 3.1). There, data were extracted from the mailbox of an authors' colleague, known to be an expert in the area of the process to discover. As usual for artful processes, the process behind the analyzed email messages was not known a priori. Therefore, we could not apply an automated comparison between the resulting workflow model and the originating process, since no definition for the originating process was available at all. Thus, the expert was requested to analyze and assess the discovered workflow model by categorizing the mined constraints. Being real data, the presence of errors in the phase of the extraction of event logs out of email messages was not tunable.

Thereafter, we created synthetic logs, where errors of different kinds were injected into event logs. Every event log was created as adhering to the specification of declarative processes comprising a single constraint at a time. For each log, i.e., a different constraint template was considered. Being known a priori the only constraint to be considered valid, when mined out of the synthetic log, we focused on the trend of its support, in order to monitor the robustness of the template w.r.t. given types of errors. We outline the results of that analysis in Section 3.2.

3.1 A real case study

As real data to conduct the experiments on, we took 6 mailbox IMAP folders containing email messages which concerned the management of 5 different European research projects (Figure 1a). Such folders belonged to a domain expert. Our aim was to use MailOfMine in order to discover the artful process of managing European research projects and validate the result, together with him.

In order to ease the revision process of the gathered results, we restricted the number of activities for the process to discover to 13. 8.998% of the total amount of email messages were considered related to the execution of an activity. The setup and the results of the inspection of email messages for extracting a log is quantitatively summarized in Table 1b. The log was passed to the control flow discovery algorithm, which returned a process model comprising c.a. 200 constraints. Each was verified to hold true within the log and associated to a support exceeding the user-defined threshold of 80%.

	Mailbox						
	1	2	3	4	5	6	Total
Messages	3523	39	844	4746	1479	60	8770

(a) The input

Activities	13	Noticeably right discovered constraints	14 (6.422%)
Traces	6	Right discovered constraints	173 (69.725%)
Events	139	Wrong discovered constraints	45 (20.642%)
Discovered constraints	218	Utterly wrong discovered constraints	7 (3.211%)

(b) Retrieved information and mined constraints

Table 1: Evaluation of MailOfMine on real data: setup and results

In order to assess the validity of the mined process, we checked every constraint with the expert. This allowed us for a quantitative evaluation.For each constraint in the list, we asked him whether it was either: *(i)* right, i.e., it made sense with respect to his experience; *(ii)* noticeably right, i.e., it not only made sense but also suggested some surprising mechanisms in the workflow; *(iii)* wrong, i.e., not necessarily corresponding to reality; *(iv)* utterly wrong, i.e., not corresponding to reality, unreasonable. The last level was assigned to quite few constraints (7 out of 218), a half of how many were considered noticeably right (14). The model is not known a priori, but the expert could classify as right or wrong a guessed constraint. Then, the analysis helped us find only true positives (TP, i.e., right or noticeably right) and false positives (FP, i.e., wrong or utterly wrong). As a matter of fact, such situation of partial knowledge of the workflow reproduces a real case, where the artful process had not ever been formalized before.

Recalling that $Precision = \frac{TP}{TP+FP}$, the algorithm was proven to obtain a *Precision* degree of 0.794 over the real case study. Table 1b summarizes the encouraging results of this real case study evaluation. More than 75% of the constraints inferred were compliant to a realistic model of the process. Figure 2 shows the trend of true positives, false positives and overall (i.e., the sum of the preceding) constraints found, scaled in percentage by their total amount, with respect to their support. The quantities on the ordinates are cumulative, i.e., they represent the sum of the values which are gained up to the current abscissa. The curves show how, as the support increases, the distance between the cumulated false positives and the true positives grow. A line puts in evidence where the relative percentage of confirmed constraints overtakes the wrong, i.e., a "breakpoint" after which the rate of hits, in terms of accepted guesses, is higher than the rate of misses, in terms of wrong guesses. Such breakpoint corresponds to a support value of 0.85 (i.e., 5% higher than the threshold established a priori), which is little enough to limit the number of true positives below that soil to less than 10%. The same graph, although, depicts that more than 90% of errors are given a support exceeding that soil as well. Thus, shifting the threshold altogether would not lead to significant improvements in the quality of the returned process. Hence, we studied the trend of support for error-injected logs, taking into account and isolating the behavior of every constraint template to different types of errors.

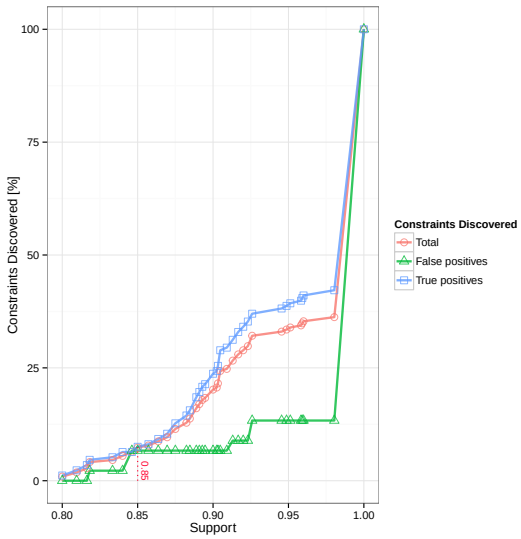


Fig. 2: The trend of the cumulative sum of constraints discovered, scaled by their total amount, w.r.t. the assigned support

3.2 Experiments over artificial error-injected logs

In order to test the robustness of MINERful with respect to the presence of errors in logs, we built an additional testing module, which injected a controlled amount of noise in the sequences of traces.

We identified three possible *types of error injection*:

1. insertion of spurious events in the log;
2. deletion of events from the log;
3. random insertion/deletion of events.

The errors were spread according to a given percentage[3]. The tester could also specify whether errors had to refer to a given activity, or not. In the latter case, every insertion or deletion was applied to an event picked each time at random.

In order to define how many errors had to be injected, and where, a *spreading policy* was requested too. It could be either:

1. to calculate the number of errors to inject w.r.t. the whole log, and distribute the error injections accordingly, or
2. to calculate the number of errors to inject w.r.t. every single trace, case by case.

In the latter case, every trace was made affected by a number of errors, computed on the basis of the number of target events in that trace. This reproduces a systematic error, taking place in every registered enactment of the process. In the former, some traces could remain untouched.

[3] In case the calculated number of errors to inject resulted in a non-integer number, the actual amount of errors was rounded up to the next integer (e.g., 0.2 was rounded to 1 error to inject).

Thereupon, we conducted an extensive analysis on the reaction of MINERful, the control flow discovery algorithm of MailOfMine, through an experiment set up as summarized in Table 2.

Activities (target) 8 (1)	Spreading policies 3
Generating constraints 18	Error types 3
Trace length $[0, 30]$	Runs per combination 50
Log size 1 000	Error injection percentage $[0, 30]$
	Total runs 167 400

Table 2: Setup of the experiments for monitoring the reaction of MINERful to the controlled error injection into logs

We created 18 groups of 9 300 synthetic logs each. Every group was generated so to comply to one constraint at a time, among the 18 templates involving a, as the implying activity, and (optionally) b, as the implied (i.e., *Participation*(a), *Uniqueness*(a), ..., *RespondedExistence*(a, b), *Response*(a, b), ...). The alphabet comprised 6 more non-constrained activities (c, d, ..., h), totalling 8. We chose a as the target activity for the injection of errors. Then, we injected errors in the synthetic logs, with all of the possible combinations of the aforementioned parameters (*(i)* insertion, deletion or random error type, *(ii)* over-string or over-collection spreading policy, *(iii)* error injection percentage ranging between 0 and 30%) and ran the control flow discovery algorithm of MailOfMine on the resulting altered logs. We collected the results and, for each of the 18 groups of logs, analyzed the trend of the support for the generating constraint. I.e., we looked at how the support for the only constraint which had to be verified all over the log lowered, w.r.t. the increasing percentage of errors injected. We also hightlighted those other constraints whose topmost computed support exceeded the value of 0.75[4], being them the most likely candidates to be false positives in the discovery.

The analysis of within-trace error-injected logs revealed to be more effective in stressing the resilience of constraints with respect to certain types of errors. In other words, it showed the structural weaknesses of constraint templates w.r.t. some types of error even for small percentages of injected errors. For instance, the support of *End*(a)'s (Figure 3) is not affected by the insertion of spurious a's in the traces (see Figure 3a), whereas it suffers from deletions of a's (Figure 3b).

In Section 2 we described the mechanism tying *MutualRelation* constraints to *forward* and *backward*-related constraints, as in the case of *AlternateSuccession* w.r.t. *AlternateResponse* and *AlternatePrecedence*. Then, here we remark that since *(i)* the support for *AlternateResponse*(a, b) remains unchanged in case of spurious inserted a's (Figure 4a), but not in case of deleted a's (Figure 4b), whilst

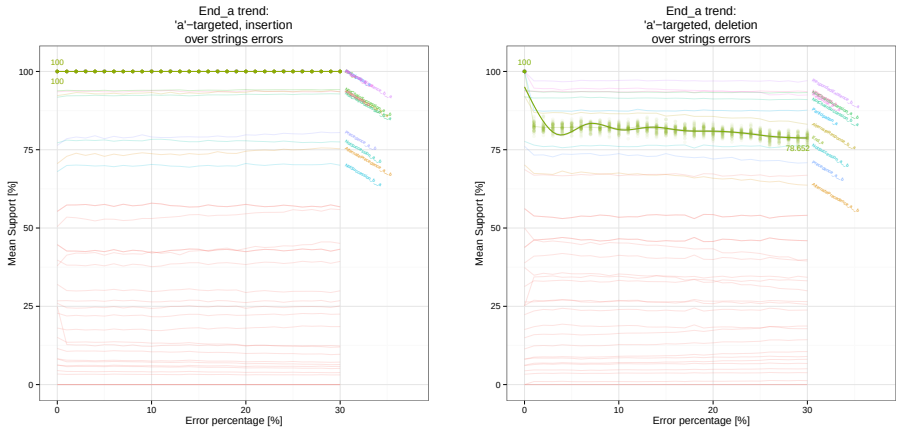
[4] We recall that assigning a constraint the support of 0.5 would be equivalent to asserting that such constraint would hold if, tossing a coin, a cross was shown in the end. Thereby, 0.75 is the least value of the topmost half of the "reliable" range.

(ii) conversely, the support for *AlternatePrecedence*(a, b) remains unchanged in case of deleted a's (Figure 4c), but not in case of inserted spurious a's (Figure 4d), *AlternateSuccession* inherits the sensitivity towards errors of both, resulting in a decreasing support for both faulty insertions and deletions of a's (Figure 5).

The analysis of over-collection error-injected logs showed smoother changes in curves, since errors are spread on a wider area of appearances, for the targeted activity. Therefore, it reveals a more realistic trend for the assessment of discovered constraints in presence of errors. We reasonably expect to have sparse errors in logs, rather than a fixed percentage of faults for every trace, as a matter of fact.

Along a branch in the constraints hierarchy (see Figure 1), we expect that the more a constraint is restrictive, the more its support decreases in terms of deviations from the expected behavior. We can prove it by evidence in, e.g., Figure 6, where the curve's slope gets steeper as we analyze the subsumed constraints along the *MutualRelation* constraints (i.e., *CoExistence*, *Succession*, *AlternateSuccession*, *ChainSuccession*).

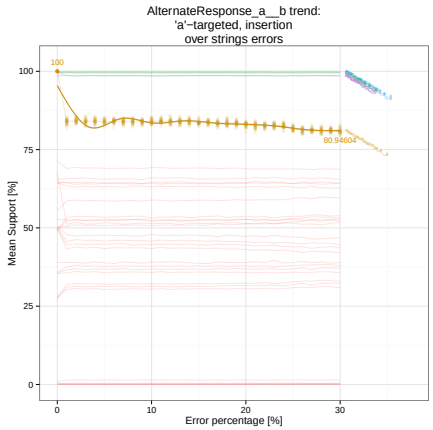
The interested reader can download the whole collection of graphs depicting the gathered results at the following address:
`http://www.dis.uniroma1.it/~cdc/code/minerful/latest/errorinjectiontestresults.zip`



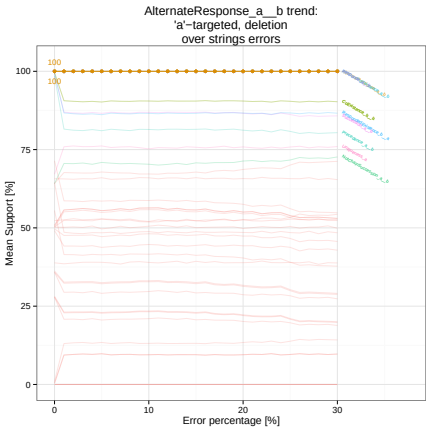
(a) The trend of the support for *End*(a), w.r.t. the percentage of spurious events inserted into every string

(b) The trend of the support for *End*(a), w.r.t. the percentage of events deleted from every string

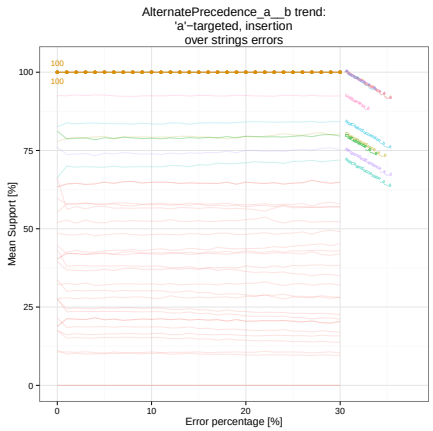
Fig. 3: The trend of the support for *End*, w.r.t. the errors injected in the log, within every trace



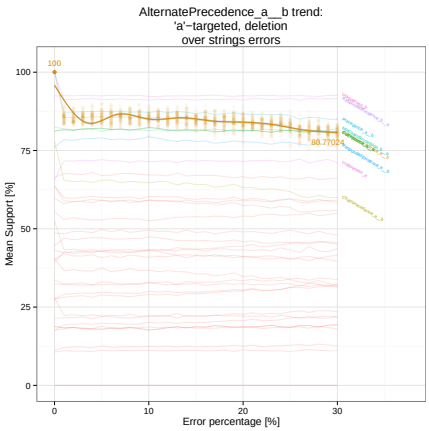
(a) The trend of the support for *AlternateResponse*(a, b), w.r.t. the percentage of spurious events inserted into every string



(b) The trend of the support for *AlternateResponse*(a, b), w.r.t. the percentage of spurious events inserted into every string

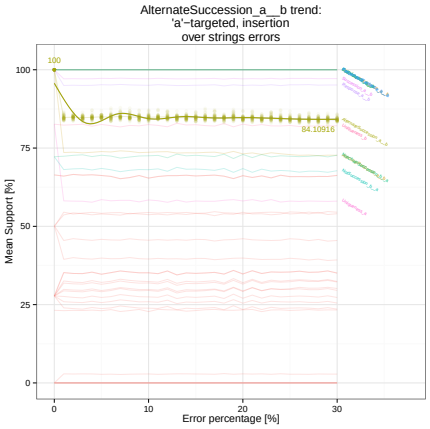


(c) The trend of the support for *AlternatePrecedence*(a, b), w.r.t. the percentage of events deleted from every string

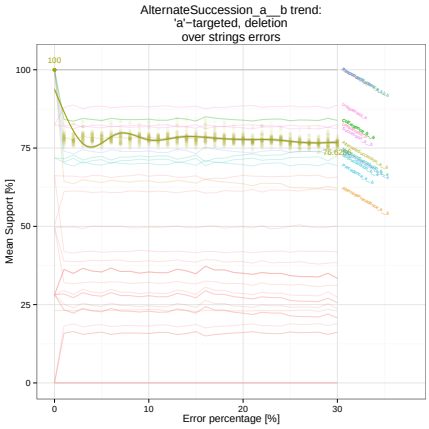


(d) The trend of the support for *AlternatePrecedence*(a, b), w.r.t. the percentage of events deleted from every string

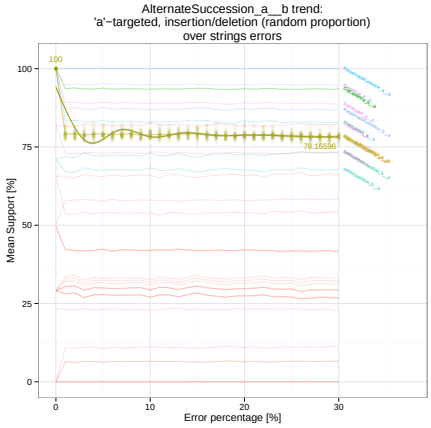
Fig. 4: The trend of the support for *AlternateResponse* and *AlternatePrecedence*, w.r.t. the errors injected in the log. The error injection policies under exam are both the insertion and deletion of a events, within each trace.



(a) The trend of the support for *AlternateSuccession*(a, b), w.r.t. the percentage of spurious events inserted into every string

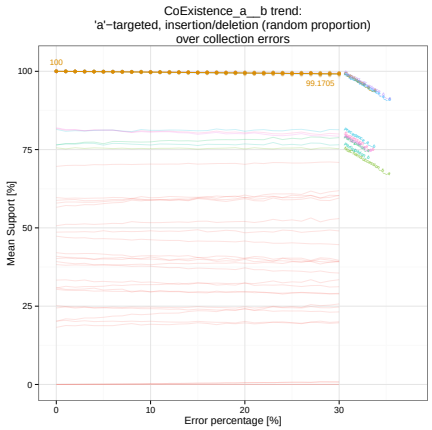


(b) The trend of the support for *AlternateSuccession*(a, b), w.r.t. the percentage of events deleted from every string

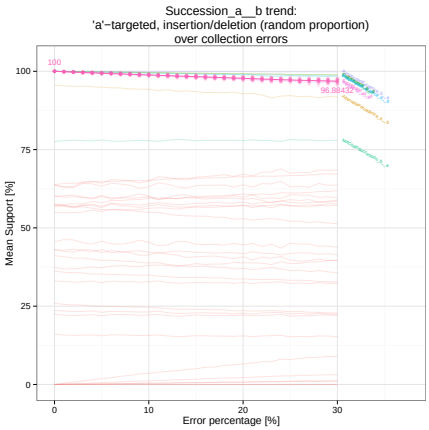


(c) The trend of the support for *AlternateSuccession*(a, b), w.r.t. the percentage of both deletions and insertions, applied to every string

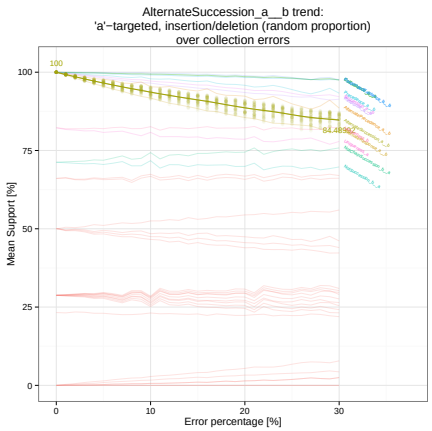
Fig. 5: The trend of the support for *AlternateSuccession*, w.r.t. the errors injected in the log, within each trace.



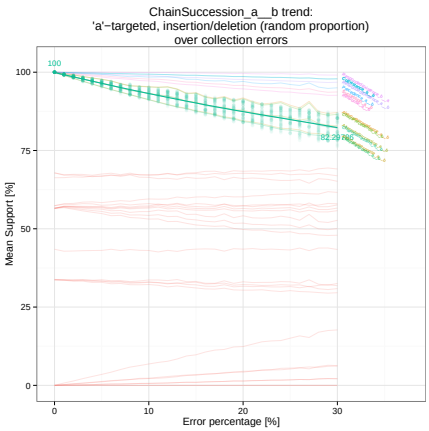
(a) The trend of the support for *CoExistence*(a, b), w.r.t. the percentage of both event deletions and insertions, spread over the whole log



(b) The trend of the support for *Succession*(a, b), w.r.t. the percentage of both event deletions and insertions, spread over the whole log



(c) The trend of the support for *AlternateSuccession*(a, b), w.r.t. the percentage of both event deletions and insertions, spread over into the whole log



(d) The trend of the support for *ChainSuccession*(a, b), w.r.t. the percentage of both event deletions and insertions, spread over into the whole log

Fig. 6: The trend of the support for the *MutualRelation* constraints, w.r.t. the errors injected in the log. The error injection policy under exam is the random insertion/deletion of a events, over the whole log.

4 Conclusions

Throughout this paper, we have analyzed the problem of discovering declarative workflows out of event logs which are affected by errors. To this aim, we first assessed the quality of a model, mined out of real data. We used a single threshold level for the estimated support of discovered constraints, in order to determine whether they could be considered valid for the mined process or not. The obtained results suggested that adjusting the level of such threshold did not considerably enhance the quality of the mined process altogether. Therefore, for each constraint in the set of Declare templates, we investigated the trend of its own estimated support with respect to the amount of errors injected into logs. By means of experiments carried out on synthetic data, we showed that the semantics of constraint templates actually affect their degree of robustness w.r.t. the presence or spurious events or the absence of expected ones in the log.

Starting from these results, we will investigate the problem of defining an automated approach for the self-adjustment of user-defined thresholds, on the basis of the nature of each discovered constraint. Intuitively, indeed, a more "robust" constraint should be considered valid in the log (and therefore for the process) if its support exceeds a higher threshold. On the contrary, the threshold should be diminished for more "sensitive" ones. We also aim at mixing such an approach with the analysis of different metrics, pertaining to the number of times an event occurred in the log. The intuition is that the more an event is frequent in the log, the less it can be considered subject to errors. Such metrics have been already considered in literature ([15]) for assessing the relevance of discovered constraints. We want to exploit them for estimating the reliability of constraints in mined processes as well.

References

1. van der Aalst, W.M.P.: Verification of workflow nets. In: Azéma, P., Balbo, G. (eds.) ICATPN. Lecture Notes in Computer Science, vol. 1248, pp. 407–426. Springer (1997).
2. van der Aalst, W.M.P.: The application of petri nets to workflow management. Journal of Circuits, Systems, and Computers 8(1), 21–66 (1998).
3. van der Aalst, W.M.P.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer (2011).
4. van der Aalst, W.M.P., ter Hofstede, A.H.M.: YAWL: yet another workflow language. Inf. Syst. 30(4), 245–275 (2005).
5. van der Aalst, W.M.P., Pesic, M., Schonenberg, H.: Declarative workflows: Balancing between flexibility and support. Computer Science - R&D 23(2), 99–113 (2009).
6. Alberti, M., Chesani, F., Gavanelli, M., Lamma, E., Mello, P., Torroni, P.: Verifiable agent interaction in abductive logic programming: The SCIFF framework. ACM Trans. Comput. Log. 9(4), 29:1–29:43 (August 2008).
7. Chesani, F., Lamma, E., Mello, P., Montali, M., Riguzzi, F., Storari, S.: Exploiting inductive logic programming techniques for declarative process mining. T. Petri Nets and Other Models of Concurrency 2, 278–295 (2009).

8. Decker, G., Dijkman, R.M., Dumas, M., García-Bañuelos, L.: The business process modeling notation. In: ter Hofstede, A.M., van der Aalst, W.M.P., Adamns, M., Russell, N. (eds.) Modern Business Process Automation, pp. 347–368. Springer (2010).
9. Di Ciccio, C., Marrella, A., Russo, A.: Knowledge-intensive processes: An overview of contemporary approaches. In: ter Hofstede, A.H., Mecella, M., Sardina, S., Marrella, A. (eds.) KiBP. vol. 861, pp. 33–47. CEUR Workshop Proceedings (06 2012).
10. Di Ciccio, C., Mecella, M.: Mining constraints for artful processes. In: Abramowicz, W., Kriksciuniene, D., Sakalauskas, V. (eds.) BIS. Lecture Notes in Business Information Processing, vol. 117, pp. 11–23. Springer (05 2012).
11. Di Ciccio, C., Mecella, M.: A two-step fast algorithm for the automated discovery of declarative workflows. In: CIDM. IEEE (04 2013).
12. Di Ciccio, C., Mecella, M., Scannapieco, M., Zardetto, D., Catarci, T.: MailOfMine – analyzing mail messages for mining artful collaborative processes. In: Aberer, K., Damiani, E., Dillon, T. (eds.) Data-Driven Process Discovery and Analysis, Lecture Notes in Business Information Processing, vol. 116, pp. 55–81. Springer (10 2012).
13. Fahland, D., van der Aalst, W.M.P.: Repairing process models to reflect reality. In: Barros, A.P., Gal, A., Kindler, E. (eds.) BPM. Lecture Notes in Computer Science, vol. 7481, pp. 229–245. Springer (2012).
14. Hill, C., Yates, R., Jones, C., Kogan, S.L.: Beyond predictable workflows: Enhancing productivity in artful business processes. IBM Systems Journal 45(4), 663–682 (2006).
15. Maggi, F.M., Bose, R.P.J.C., van der Aalst, W.M.P.: Efficient discovery of understandable declarative process models from event logs. In: Ralyté, J., Franch, X., Brinkkemper, S., Wrycza, S. (eds.) CAiSE. Lecture Notes in Computer Science, vol. 7328, pp. 270–285. Springer (2012).
16. Maggi, F.M., Mooij, A.J., van der Aalst, W.M.P.: User-guided discovery of declarative process models. In: CIDM. pp. 192–199. IEEE (2011).
17. Mendling, J., Neumann, G., van der Aalst, W.M.P.: Understanding the occurrence of errors in process models based on metrics. In: Meersman, R., Tari, Z. (eds.) CoopIS. Lecture Notes in Computer Science, vol. 4803, pp. 113–130. Springer (2007).
18. Mendling, J., Reijers, H.A., van der Aalst, W.M.P.: Seven process modeling guidelines (7PMG). Information & Software Technology 52(2), 127–136 (2010).
19. Mendling, J., Reijers, H.A., Cardoso, J.: What makes process models understandable? In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM. Lecture Notes in Computer Science, vol. 4714, pp. 48–63. Springer (2007).
20. Pesic, M.: Constraint-based Workflow Management Systems: Shifting Control to Users. Ph.D. thesis, Technische Universiteit Eindhoven (10 2008), `http://repository.tue.nl/638413`
21. Pesic, M., Schonenberg, H., van der Aalst, W.M.P.: Declare: Full support for loosely-structured processes. In: EDOC. pp. 287–300. IEEE Computer Society (2007).
22. Pesic, M., Schonenberg, M.H., Sidorova, N., van der Aalst, W.M.P.: Constraint-based workflow models: Change made easy. In: Meersman, R., Tari, Z. (eds.) CoopIS. Lecture Notes in Computer Science, vol. 4803, pp. 77–94. Springer (2007).
23. Warren, P., Kings, N., Thurlow, I., Davies, J., Buerger, T., Simperl, E., Ruiz, C., Gomez-Perez, J.M., Ermolayev, V., Ghani, R., Tilly, M., Bösser, T., Imtiaz, A.: Improving knowledge worker productivity - the Active integrated approach. BT Technologiy Journal 26(2), 165–176 (2009).

Development of a knowledge base for enabling non-expert users to apply data mining algorithms

Roberto Espinosa[1], Diego García-Saiz[2], Marta Zorrilla[2], Jose Jacobo Zubcoff[3], Jose-Norberto Mazón[4]

[1] WaKe Research, Universidad de Matanzas "Camilo Cienfuegos", Cuba
roberto.espinosa@umcc.cu
[2] MatEsCo, Universidad de Cantabria, Santander, Spain
{diego.garcias,marta.zorrilla}@unican.es
[3] WaKe Research, Dept. Ciencias del Mar y Biología Aplicada, Universidad de Alicante, Spain
jose.zubcoff@ua.es
[4] WaKe Research, Dept. Lenguajes y Sistemas Informáticos, Universidad de Alicante, Spain
jnmazon@dlsi.ua.es

Abstract. Non-expert users find complex to gain richer insights into the increasingly amount of available data. Advanced data analysis techniques, such as data mining, are difficult to apply due to the fact that (i) a great number of data mining algorithms can be applied to solve the same problem, and (ii) correctly applying data mining techniques always requires dealing with the data quality of sources. Therefore, these non-expert users must be informed about what data mining techniques and parameters-setting are appropriate for being applied to their sources according to their data quality. To this aim, we propose the construction of an automatic recommender built using a knowledge base which contains information about previously solved data mining tasks. The construction of the knowledge base is a critical step in the recommender design. We propose a model-driven approach for the development of a knowledge base, which is automatically fed by a Taverna workflow. Experiments are conducted to show the feasibility of our knowledge base as a resource in an online educational platform, in which instructors of e-learning courses are non-expert data miners who need to discover how their courses are used in order to make informed decisions to improve them.

Keywords: knowledge base, data mining, recommenders, meta-learning, model-driven development

1 Introduction

The increasing availability of data is a great opportunity for everyone to take advantage of their analysis. Physicians in hospitals, lawyers in the law business, teachers in high schools or universities, or even regular citizens, would be interested in applying advanced data analysis techniques to make informed decisions in their daily life. Importantly, data mining is one of the most prominent technique to discover implicit knowledge patterns, thus gaining richer insights into data [16].

However, non-expert users may find complex to apply data mining techniques to obtain useful results, due to the fact that it is an intrinsically complex process [17, 23] in which (i) a great number of algorithms can be applied to solve the same problem with different outcomes, and (ii) correctly applying data mining techniques always

requires a lot of manual effort for preparing the data sets according to their quality. Consequently, data mining requires the know-how of an expert in order to obtain reliable and useful knowledge in the resulting patterns. Democratization of data mining therefore requires relying on knowledge about what data mining techniques and parameters-setting are appropriate for being applied to their sources according to their data quality.

User-friendly data mining [14] is a step forward to this democratization, since it fosters knowledge discovery without mastering concepts and data mining techniques. To realize user-friendly data mining, in this paper we propose a model-driven approach for the development of a data-mining knowledge base. It contains information about the behavior of data mining algorithms in presence of one or several data quality criteria and intrinsic characteristics of the data sets. This information comes from a set of experiments automatically obtained by means of a Taverna workflow in order to be easily replicated as well as enabling the extension of the knowledge base. A model-driven development approach is proposed in order to obtain the information extracted from our Taverna workflow in a standard manner and automatically generating the knowledge base as a set of models.

The process of building the data-mining knowledge base starts when a dataset is selected as new source in our Taverna workflow. Then, a set of data quality criteria are measured when some mining algorithms are applied to the dataset. This information is stored in a model which is automatically created by using a model-driven approach. An overview of our approach is shown in Figure 1.

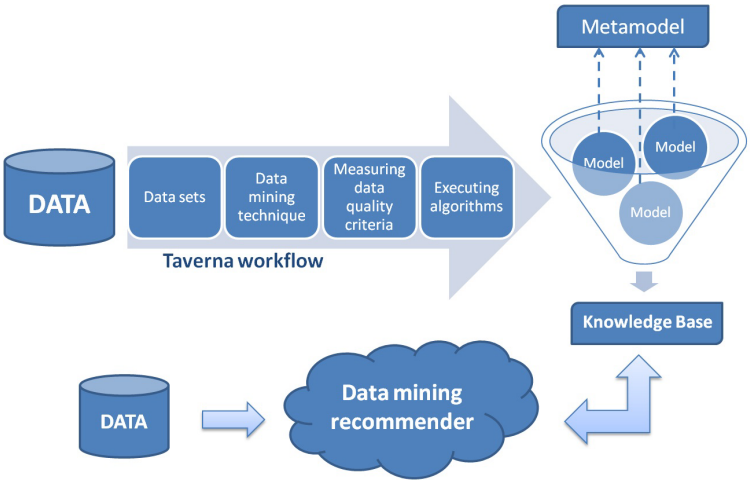


Fig. 1. Developing our knowledge base.

It is worth noting that our knowledge base can be used (i) directly, by non-expert data miners that have certain expertise in data management; or (ii) indirectly, by using a kind of "recommender" that query the knowledge base to guide non-expert data miners by suggesting the best algorithm to be applied to the data, or even to guide experts data miners by suggesting, for example, the algorithms they should use at the beginning of their study.

Some experimentation is conducted in order to evaluate our knowledge base as a resource for a non-expert data miner in an online educational context: instructors

of e-learning courses are non-expert data miners who need to discover whom and how their courses are used in order to improve them. Data mining is being profusely used [21] in the educational context as consequence of the rapid expansion of the use of technologies in supporting learning, not only in established institutional contexts and platforms, but also in the emerging landscape of free, open, social learning online. Although there are tools as ElWM [29] which help instructors to analyse their virtual courses, a knowledge base as proposed here will become a crucial resource for designing a recommender that help instructors (as non-expert data miners) in applying the right data mining algorithm on their data sets and to extract conclusions oriented to improving the teaching-learning process.

This paper is therefore a step forward to realize the user-friendly data mining. Specifically, the contributions of this paper are as follows:

1. A metamodel that contains those useful concepts for representing models with information about data mining experiments: data's sources metadata, results of data mining algorithms, and values of data quality criteria.
2. A knowledge base as a repository of models that contains the data mining information.
3. A Taverna workflow for providing a mechanism to obtain all the information to automatically create our knowledge base.
4. A set of experiments addressed to build a recommender are shown as proof of feasibility of our approach

The remainder of this paper is structured as follows: an overview of the related work is presented in section 2. Our knowledge base is introduced in section 3, while the conducted experiments are described in section 4. Finally, conclusions and future work are sketched in section 5.

2 Related work

The data mining algorithm selection is at the core of the knowledge discovery process [5]. Several data mining ontologies have been developed to provide adequate knowledge to help in this selection. For example, OntoDM [18] is a top-level ontology for data mining concepts that describes basic entities aimed to cover the whole data-mining domain, while EXPO ontology [22] is focused on modeling scientific experiments. A more complete ontology is DMOP [9] which not only describes learning algorithms (including their internal mechanisms and models), but also workflows. Furthermore, a large set of data mining operators are described in the KD ontology [28] and the eProPlan ontology [12].

Regarding data mining workflows, the KDDONTO ontology [3] aims at both discovering suitable KD algorithms and describing workflows of KD processes. It is mainly focused on concepts related to inputs and outputs of the algorithms and any pre and post-conditions for their use. Also, the Ontology-Based Meta-Mining of Knowledge Discovery Workflows [10] is aimed at supporting workflow construction for the knowledge discovery process. Moreover, in [25] authors propose a specific ontology to describe machine learning experiments in a standardized manner for supporting a collaborative approach to the analysis of learning algorithms (further developed in [24]).

There are some projects that allow scientific community to contribute with their experimentation in improving the knowledge discovery process. The Machine Learning Experiment Database developed by University of Leuven [2] offers a Web tool to store the experiments performed in a database and query it. The e-LICO project funded

by the Seventh Framework Programme [8] has developed a knowledge-driven data mining assistant which relies on a data mining ontology to plan the mining process and propose ranked workflows for a given application problem [10].

Unlike our proposal, both projects are oriented to support expert data miners. Our knowledge base would help naive data miners and non-experts users to have a kind of guidance about which techniques can or should be used and in which contexts.

Furthermore, although ontologies used in the aforementioned approaches are very useful for providing semantics, they lack mechanisms for automating the management (and interchange) of metadata, such as metamodeling [19]. Under the model-driven umbrella, and according to [13], a model is a "description of (part of) a system written in a well-defined language, while a well-defined language is a language with well-defined form (syntax), and meaning (semantics), which is suitable for automated interpretation by a computer". Therefore, on the one hand, a model must focus on those important parts of a system, thus avoiding superfluous details. On the other hand, well defined languages can be designed by means of metamodeling [1], which provides the foundation for creating models in a meaningful, precise and consistent manner. Therefore, metamodelling provides a common structure for storing the most relevant information in models, thus avoiding interoperability and compatibility problems. For example, having a metamodel allows us to specify data coming from different DBMS in a model which can be easily used as input data set for data mining experiments.

Our aim in this work is creating a metamodel inspired by the aforementioned data mining ontologies that allows us to create a set of models to create a knowledge base for data mining. Moreover, in previous experiments we have demonstrated the influence of data quality in the results obtained when applying techniques of data mining [4].

3 Model-driven approach for knowledge base development

Our knowledge base brings the results on executing data mining processes on many data sets. It can be therefore used as a resource to keep information about the behavior of different data mining algorithms with regard of the data sources quality and general characteristics of the data set. Collected information can be useful for supporting non-expert users in a decision making process and which is the best data mining algorithm to apply according to the available data. To this aim, our knowledge base contains the following information:

Information from input data sets. Metadata from the data sets must be known, as number of attributes and instances, as well as the corresponding data types.

Results when applying a data mining algorithm. Some information related to the execution of a data mining algorithm is acquired: data mining technique being executed, predicted attribute and their results.

Data quality criteria. Several quality criteria from the data sets must be measured. Quality criteria can be related to data sets (e.g. percentages of null values), as well as fields (e.g. field correlation).

3.1 Scientific workflow for the development of our knowledge base

The development of our data mining knowledge base is driven by the development of a scientific workflow. This workflow is in charge of (i) collecting all the required information for our knowledge base (as previously stated), (ii) creating the knowledge

base, and (iii) implementing a recommender for data mining algorithms based on our knowledge base.

Scientific workflows are largely recognized as useful paradigms to describe, drive, and share information about experiments[5]. Specifically, Taverna Workbench[6] is used in our approach. Taverna is part of myGrid project[7], that aims to produce and use a suite of tools designed to allow international communities to publish and share information.

Our workflow has as a main objective the datasets processing in order to create models to conform the knowledge base. To this end, the workflow begins with the loading of the data source (e.g. *.arff* files[8]) on which will be applied a set of data mining algorithms. Then, the type of data mining technique must be executed[9]. Next step is about to obtain a predicted attribute (usually the last column). Subsequently, in order to have a visual output in the workflow, expert user can select the resulting algorithm values (e.g. correctly classified instances, mean absolute error, precision, etc.), although all these results are part of the obtained model, and all data mining algorithms are executed, leading to a result set. Simultaneously, the workflow measures the quality criteria values of the data source according to some quality criteria. The workflow can be run manually or configured by command line.

Once required information is acquired, the knowledge base is generated as explained in the following subsection.

3.2 Generating a data mining knowledge base

Our knowledge base aims to represent in a structured and homogeneous manner all the necessary data mining concepts. Following the model-driven paradigm [1], our knowledge base is uniform and automatically created as a repository of models that conforms to a metamodel for representing the output information of our Taverna workflow. Once, the knowledge base is obtained the non-expert miner could use it to evaluate the real dataset in order to obtain the adecuated predicted model having in account the dataset features.

The aim of our metamodel is being as generic as possible. Therefore, any data related to the aforementioned information about data mining experiments (metadata of data sources, results of data mining algorithms, and values of data quality criteria) is adequately represented in a model. Our models are not restricted to a certain quality criteria, since the metamodel support creating new quality criteria in each model as required. The definition of our metamodel (see Fig. 3) is based on an analysis of several ontologies (see Section 2):

DMKBModel. This is the main class that contains the other useful elements for representing a Data Mining Knowledge Base (DMKB). The `DMKBModel` class allows the specification of a model in which the following information can be stored: input data sets, metadata, data mining algorithms, parameter-setting, data mining results generated when the Taverna workflow is executed, and data quality criteria.

[5] http://en.wikipedia.org/wiki/Scientific_workflow_system
[6] http://www.taverna.org.uk/
[7] http://www.myGrid.org.uk
[8] Attribute-Relation File Format (ARFF), a file format used by the data mining tool Weka [6] to store data.
[9] Our Taverna workflow was designed to be useful for any mining technique, but in this paper we only consider classification techniques.

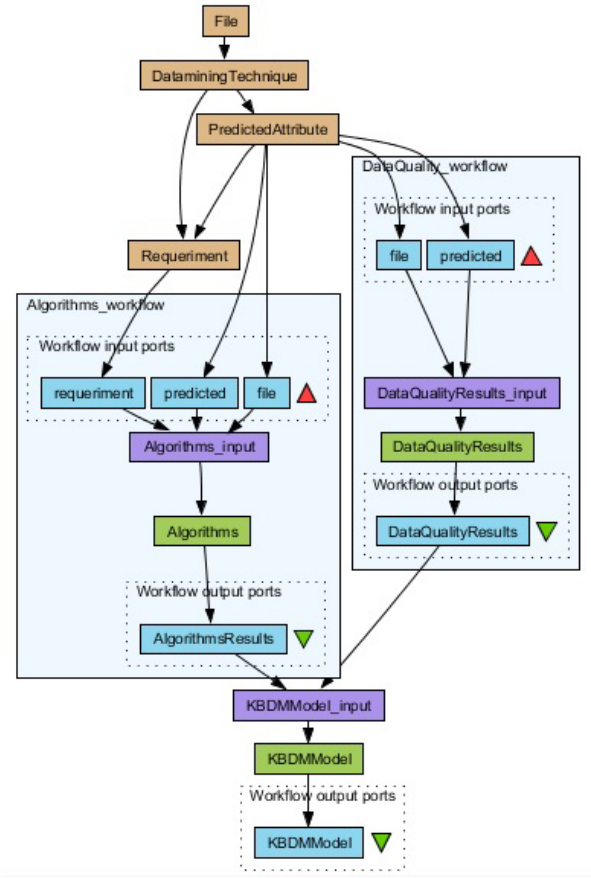


Fig. 2. Our Taverna workflow.

DataSet. It describes data sets used for generating the information included in the knowledge base. Each `DataSet` is composed of different fields. Also, each data set contains a category and a set of metadata.

Field. It represents a piece of data contained in the `DataSet`. This piece of data is identified by a name. Also, the kind of field must be defined (by means of an enumeration called `FieldKind`) and its type (by means of an enumeration called `FieldType`). This class contains a set of data quality values that are related to the field.

FieldKind. It is an enumeration class for defining the general kind of values that the field instances may have (continuous, categorical or mixed).

FieldType. It is an enumeration class for representing the type of each `Field` (numeric, date, nominal or string)

DataMiningResults. This class represents values of measures for each data set after executing an algorithm (e.g. accuracy).

Algorithm. This class represent information about executed data mining algorithms. Each algorithm belongs to a specific technique. (e.g. *NaiveBayes*, *J48*, *RandomTree* or *Adaboost*).

Parameter. It is a class that represents values of initial parameters when executing an algorithm. This class contains the name of the parameter and a value.

Technique. This class defines a set of existing data mining techniques (e.g. a tree, a probability matrix, etc.). It contains a subgroup attribute in case that the algorithm requires to be further classified.

ProblemKind. It defines the different kinds of problem with which the user need is satisfied (e.g. classification, prediction, clustering, etc.).

DataQualityCriteria. It is an abstract class that represents information related to the different criteria that can be presented either in a `DataSet` (`DatasetDataQualityValue`) or in each `Field` (`FieldDataQualityValue`). For each data quality criteria, a `ComputationMode` is defined to described how it is calculated (e.g. Pearson correlation method), and a `MeasuringUnit` that represent the corresponding unit of measure.

DatasetDataQualityValue This class inherits from the `DataQualityCriteria` class and defines data quality value criteria for a `Dataset`.

FieldDataQualityValue It inherits from the `DataQualityCriteria` class and represents a value for specific `Field` class.

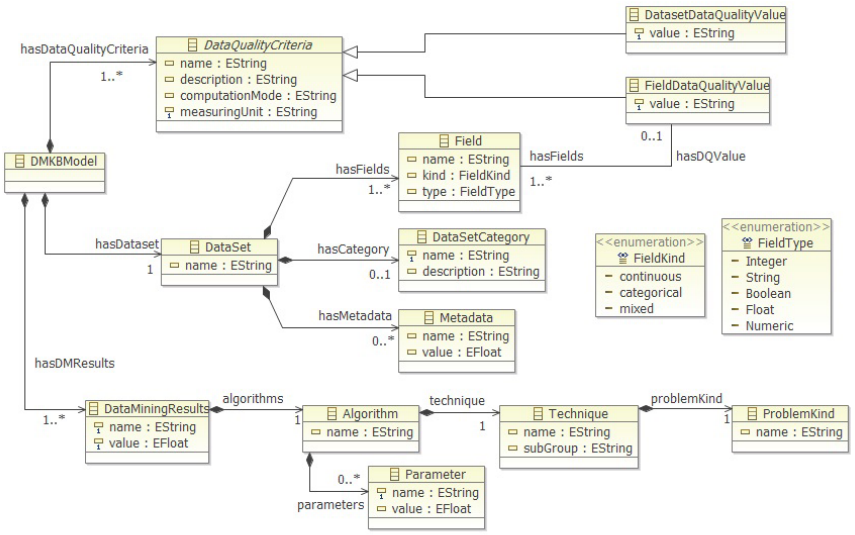


Fig. 3. Our metamodel for representing our data mining knowledge base.

As aforementioned, our Taverna workflow is in charge of handling the model-driven generation of the data mining knowledge base from the acquired information.

When a dataset is processed, all the acquired information is saved in a model conforming to the metamodel presented in Fig. 3. A set of transformations has been developed for creating the models that are integrated in the knowledge base. These transformation are executed in Taverna by means of a Web service.

Our model-driven approach is built on top of the Eclipse Framework[10], which is an open source project conceived as a modular platform which can be extended in order to add features to the development environment. Specifically, transformation tasks for

[10] http://www.eclipse.org

generating models have been supported with the use of Java facilities provided by the Eclipse Modeling Framework (EMF)[11]. The Java code in listing 1.1 shows an excerpt of the transformation in charge of creating a model within the knowledge base. For each of the data mining algorithms executed by the workflow, the following classes are generated: `DataMiningResult`, `Algorithm`, `Technique`, and `ProblemKind`; as well as the required existing relationships among them: `hasDMResults`, `algorithms`, `technique`, and `problemKind`. Finally, the model (represented by means of a XMI file) is created.

```
for (int i = 0; i <= First.listaResAlg.size()-1;
i++)
    {
        DataMiningResults dmr =kbf.createDataMiningResults();
        dmr.setName(First.listaResAlg.get(i).requirementName);
        dmr.setValue(First.listaResAlg.get(i).value);
        Algorithm alg= kbf.createAlgorithm();
        alg.setName(First.listaResAlg.get(i).algName);
        Technique tec=kbf.createTechnique();
        tec.setName(First.listaResAlg.get(i).technique);
        tec.setSubGroup(First.listaResAlg.get(i).subgroup);
        ProblemKind pk=kbf.createProblemKind();
        pk.setName(probKind);
        alg.setTechnique(tec);
        tec.setProblemKind(pk);
        dmr.setAlgorithms(alg);
        model.getHasDMResults().add(dmr);
    }
    ResourceSet rs = new ResourceSetImpl();
    rs.getResourceFactoryRegistry().getExtensionToFactoryMap().put("xmi", new XMIResourceFactoryImpl());
    Resource resource = rs.createResource(URI.createFileURI("ouput_generated/" + ds.getName() + ".xmi"));
    resource.getContents().add(model);
```

Code 1.1. Segment of Java code to create a model.

Fig. 4 shows a sample `DMKBModel` generated by using our approach. It can be observed some of the elements that conform it (e.g. Dataset, Fields, FieldDataQuality, DatasetDataQuality and DataMiningResults, which refers to the number of correctly classified instances achieved by the Decision Table algorithm for the comp2class dataset, in this case 305).

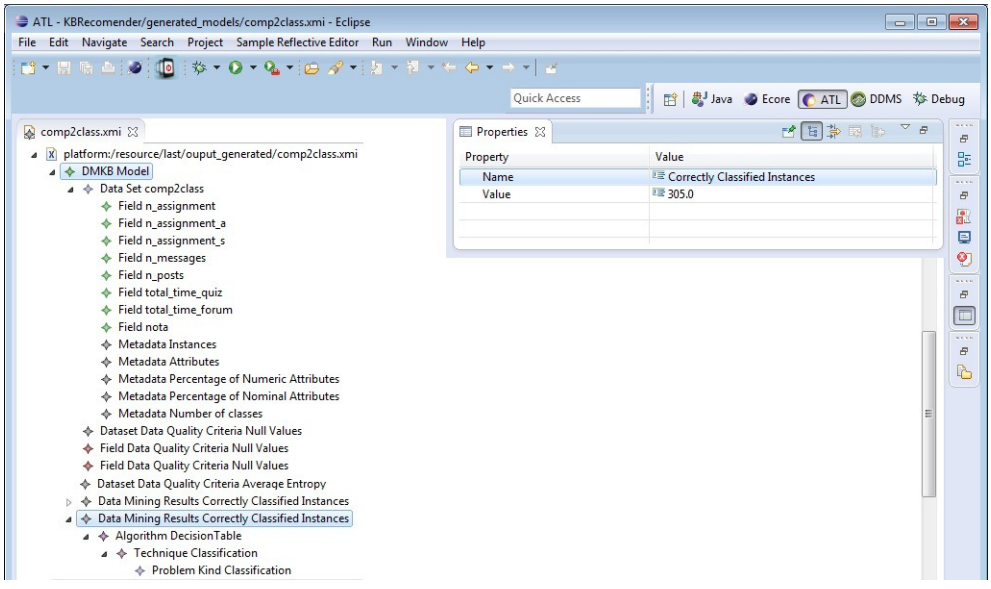


Fig. 4. Sample model of comp2class data set.

[11] http://www.eclipse.org/emf

Our knowledge base is composed by the set of models obtained after running the Taverna workflow for each input data set. These will be the data source which allows us to build our recommender.

3.3 Recommender system

A recommender system takes as input a collection of cases, each belonging to one of a small number of classes and described by its values for a fixed set of attributes, and output a classifier that can accurately predict the class to which a new case belongs (ref [25]). To create this recommender, some different classification algorithms and features can be used. In our case we used as input the number of instances, number of attributes, percentage of nominal attributes, percentage of numerical attributes, percentage of null values, grade of data set balance and algorithm name. We chose these features due to their strong influence in the accuracy which the recommender can achieved.

4 Experimental evaluation

Our approach has been evaluated in the e-learning domain by carrying out a experiment. The methodology followed comprises the steps listed below:

1. Selection of courses and data extraction from e-learning platforms.
2. Generation of 96 data sets as described in Sect. 4.1
3. Building of 1152 classification models from the application of 12 classification algorithms on 96 out of 99 data sets. The rest were used for testing.
4. Extraction of meta-features of each data set
5. Creation of data sets with the meta-features of each data set adding as class attribute the algorithm or algorithms which achieved the highest accuracy.
6. Building of a recommender of algorithms from our data sets with the meta-features chosen. We rely on meta-learning to build our recommender since this technique has been demonstrated suitable to assist users to choose the best algorithm for a problem at hand [11, 26].
7. Evaluation of our recommender in terms of number of times that its answer matches the algorithms that better classify the data set

In what follows, we describe the data sets and classifiers used in our experiment, along the process of building our knowledge base. Next, we explain the building of our recommender in order to show the feasibility of our proposal.

4.1 Data sets description

In our experiments, we used data from eight courses hosted in e-learning platforms at University of Cantabria (Spain): (i) one course, entitled "Introduction to multimedia methods" offered in three academic years (2007-2010) with 70 students enrolled in average and hosted in the *Blackboard e-learning platform*; (ii) seven computer science courses taught in the 2007-2008 academic year with a total of 432 enrolled students and hosted in the *Moodle Learning Content Management System*; (iii) six courses oriented to train transversal skills imparted during the first semester of 2013 with a range from 20 to 126 learners per course, also hosted in Moodle; and (iv) a semi-presential course entitled "Mathematics for economists" with 465 students enrolled.

Training data sets We defined 23 data sets with information extracted from platforms logs. Each instance in every data set represents the activity of a student in an academic year together with the final mark obtained in the course. Two different groups of data sets are considered: the training data set (used to generate the experiments to feed our knowledge base), and the test data sets (used to evaluate the recommender).

In order to have enough data sets for our experimentation, and taking into account generally data from virtual learning environments are clean, we built new data sets performing some controlled perturbations to the original datasets. The new data sets have the quality degraded, which allow us to assess if the meta-features chosen are suitable for this purpose. Furthermore, as the process to be performed by the expert should be about a monitored data set which allow validating the behavior of the algorithms under variations of the quality of data.

We generated 96 data sets from them. First we created 3 data sets with data from multimedia course establishing the class attribute with values pass or fail, and another one as the union of these three. The same process was carried out with the programming course, the "Mathematics for economists" course and the transversal courses. Next, we generated 4 discretised data sets from the previous bi-class data sets using PKIDiscretize from Weka, and 4 data sets more but these partially discretised. Besides, we created two data sets with 4 classes (fail, pass, good, excellent) and one with 5 classes (drop-out, fail, pass, good, and excellent). These are our 23 original data sets whose main features are shown in Table 1. Data sets numered from 1 to 11 correspond to the "Introduction to multimedia methods", those from 12 to 15 correspond to the computer science courses, data set 16 and 17 are from the "Mathematics for economists" course and finally data sets numbered from 18 to 23 correspond to the transversal courses.

Then, we generated 72 data sets by adding to the first eighteen data sets from Table 1 a 10, 20, 30 and 40% of missing values. And finally, we created 4 data sets more by applying SMOTE algorithm on 2 of our original data sets with the following proportion of balancing class: 80-20%, 85-15%, 70-10% and 90-10%.

Test data sets Our test data sets are described in Table 2. As can be observed, we chose three data sets with different meta-features: the first one contains the activity carried out by the students in the 2009-2010 academic year in the "Introduction to Multimedia" course (mult2class2010), it is bi-class and all attributes except the class, are numerical; the second one, collects the activity performed in the three editions of Multimedia course degraded with a 10% of missing values (multGlobalActivity); and finally, the third one gathers data from the six transversal courses mentioned above (tranversalDS) in an unique file. It is bi-class, balanced, without structural nulls, with 2 nominal and 4 numerical attributes.

They were used to evaluate the feasibility of our knowledge base for building a classifier which helps the end-user in the selection of the best algorithm.

4.2 Classifiers used in the experiment

Due to the existence of different classification algorithms, 12 different classifiers provided by Weka (trees, rules, bayesian, lazy and ensemble) were executed on the training data sets in order to feed the knowledge base. These classifiers were selected taking into account the most frequently used data mining algorithms [27] and those classifiers used in some previous works about prediction of students performance with

Table 1. Original data sets description

Name	# Instances	# Attributes	# numerical Att.	# of nominal Att.	# of classes
dataset1	64	13	13	0	2
dataset2	65	11	11	0	2
dataset3	193	22	22	0	2
dataset4	193	22	22	0	4
dataset5	193	22	22	0	5
dataset6	193	22	0	22	2
dataset7	193	22	15	7	2
dataset8	64	13	0	13	2
dataset9	64	13	7	6	2
dataset10	65	11	0	11	2
dataset11	65	11	5	6	2
dataset12	438	14	14	0	2
dataset13	438	14	14	0	4
dataset14	438	14	0	14	2
dataset15	438	14	5	9	2
dataset16	465	6	0	6	2
dataset17	465	6	2	4	2
dataset18	38	4	0	4	2
dataset19	126	5	0	5	2
dataset20	28	4	0	4	2
dataset21	44	3	0	3	2
dataset22	67	6	0	6	2
dataset23	67	5	0	5	2

Table 2. Description of tests data sets

Name	# instances	# attributes	# numerical att.	# nominal att.	# classes	% missing	class balance
mult2class2010	64	18	18	0	2	0	quite_unbalanced
multGlobalActivity	193	4	4	0	2	0	balanced
tranversalDS	304	6	6	4	2	0	balanced

which we obtained the best results [7, 29]: *J48*, *SimpleCart*, *RandomForest*, *NaiveBayes*, *BayesNetwork*, *Jrip*, *Ridor*, *OneR*, *NNge*, *DecisionTable*, *K-NN*, and *Adaboost*.

4.3 Meta-features

The meta-features used in this experimentation can be classified in three groups: general, quality-related and based on information theoretic features. In particular, we selected the number of attributes and instances in the data set, the number of categorical and numerical attributes, the type of data in the data set (numeric, nominal or mixed) and the number of classes. Regarding quality, we chose completeness (percentage of null values) and finally, we used class entropy in order to establish if the class was balanced or not. We defined three possible values for this attribute: balanced, quite unbalanced, highly unbalanced.

Next, we explain how was calculated these two last meta-features.

Missing values Structural null values are considered [15]. This kind of null value does not imply that value is not known, but not applicable in certain context. Given that we consider clean our sources of data, given the existence of a null value is considered as a structural null value. The percentage of missing values has been computed by means of $numberofmissingvalues/(numberofattributes * numberofinstances)$.

Balance The unbalanced class criteria is a measure which indicates how unbalanced the class attribute is. Data stored in a certain column are balanced if the numbers of different values representing each different instance are similar, i.e., a similar number of instances are expected for each value. For a two class data set: if balanced of classes is 60-40 or less, then the data set is balanced, else if it is higher than 60-40 but lesser than 80-20, then the data set is quite unbalanced, in other case the data set is highly unbalanced. For multi class data sets (more than 2 classes), the class is highly unbalanced if some of the classes appears more than double than the others. To know how balanced data are, a method that returns the *Chi-square* for each column has been developed. Then, a statistic *Chi-square* test is performed to know if the instances are uniformly distributed. The *null* hypothesis is that all positions have similar number of instances. Then, the data would be uniformly distributed. The alternative hypothesis states that they are different. The level of significance (the point at which one can determine with 95% of confidence that the difference is not due to chance alone) is set at 0.05. The *Chi-square* formula is as follows:

$$\chi^2_{obs} = \sum_{i=1}^{n} \frac{(f_i - np_i)^2}{np_i}$$

where

- χ^2_{obs}:
- f_i: number of observed frequencies.
- p_i: number of expected frequencies.
- n is the number of categories to be considered.

4.4 Generating the knowledge base

Our knowledge base was fed with results of the training data sets. Each one of the classifiers enumerated in Section 4.2 was applied to the 96 training data sets described in Section 4.1. Results were stored in the knowledge base, together with their corresponding meta-features described in Section 4.3. This means that 1152 different models ($96 * 12$) were generated.

4.5 Results

The knowledge base is used by a recommender for selecting the best classifier for an input test data set. Therefore, the goal of this experiment is twofold: on one hand, knowing if the generated knowledge base supports the recommender in its task, and on the other hand, evaluating the goodness of our recommender.

Before knowing which are the best classifiers for each of the test data sets, we performed a clustering process using kMeans on the meta-features of the training data sets in order to discover if there were well defined patterns that we could remark. In table 3 we show the results of the 5 clusters obtained. As can be observed, cluster0 collects the data sets with a high number of instances and the nominal attributes and null instances. Cluster1 contains those data sets with the lowest number of instances and a high number of numerical attributes. Cluster2 and cluster4 are very similar, both with a high number of instances and a 100% of numerical attributes, but differ in the degree of balance, cluster2 gathers quite unbalanced instances and cluster4, highly unbalanced instances. Finally, cluster3 contains instances with a high number of attributes and the highest number of nominal values. This analysis shows that we have a suitable collection of data sets, that means, it is representative enough.

Table 3. Metadata clustering

Characteristics	cluster0	cluster1	cluster2	cluster3	cluster4
numInstances	438	119.54	512.86	147	401.37
numAtt	14	16.93	14	19	20.11
nominalAtt	85.5	8.68	0	93.29	0
numeicalAtt	15	91.07	100	6.57	100
missingValues	19.62	16.24	12.64	11.19	16.54
is_balanced	QuiteBalanced	Balanced	QuiteBalanced	Balanced	HighlyBalanced

Next, we built classifiers for our test data sets in order to know which one is the technique that best classifies each one. So that, we applied the 12 selected classifiers to the test data sets and these were ranked according to its accuracy. The best algorithms of this ranking are shown in Table 4. The table must be read as follows: the classifier which obtains the best accuracy for the *mult2class2010* data set is *NaiveBayes*, which is followed by *RandomForest* and *NNge*, and quite far by *KNN*, *J48* and *BayesNet*.

Table 4. Ranking of test data sets when applying classifiers.

Data set	Algorithm	Rank	Accuracy
mult2class2010	NaiveBayes	1	85.9375
	RandomForest	2	82.8125
	NNge	2	82.8125
	kNN	3	79.6874
	J48	4	78.125
	BayesNet	4	78.125
multGlobalActivity	BayesNet	1	84.0206
	SimpleCart	2	83.5052
	DecisionTable	2	83.5052
	J48	3	82.9897
	Jrip	3	82.9897
transversalDS	J48	1	86.1963
	kNN	2	85.5828
	JRip	3	84.3558
	RandomForest	4	823.7423
	SimpleCart	5	83.4653

Next, we built two different recommenders using *J48* and *NaiveBayes* algorithms, respectively. The meta data set used contained 111 instances, that means, one instance with the meta-features of each data set together the best algorithm which performed the classification task. Since some data sets were classified by more than

one algorithm with the same accuracy, these appears twice, once with each algorithm. The data set considered for this task contained the instances of our knowledge base corresponding to the four classifiers that achieved more times the better results, which are (*NaiveBayes*, *J48*, *Jrip* and *BayesNet*).

The recommendation given for each data set by each recommender is shown in Table 5. As can be observed, the recommender based on *J48* recommends, for multGlobalActivity data set, one of the best classifiers, *Jrip*; and the best one for mult2class2012 and transversalDS datasets, *NaiveBayes* and *J48* respectively. The recommender based on*NaiveBayes* recommends one of the best classifiers for the multGlobalActivity dataset, *J48*, and for transversalDS data set, *Jrip*. Thus, we conclude that these recommenders select one of the best classification algorithms.

Table 5. Recommender results.

Dataset	J48 Recommendation	NB Recommendation
multGlobalActivity	Jrip	J48
mult2class2010	NaiveBayes	Jrip
transversalDS	J48	Jrip

Finally, we built another recommender, in this case, we used the 12 classifiers described in Section 4.2. Results are shown in Table 6. In *multGlobalActivity* data set, the recommender based on *J48* recommends to use *Jrip*, which is one of the best algorithms to classify this data set. Moreover, for transversalDS data set, it recommends the best classifier, *J48*. The recommender based on *NaiveBayes* also recommends one of the best algorithms for *mult2class2010*: *RandomForest*. However, the results are worse than in previous experiment in which we only considered four classifiers for our predictive attribute. This happens because, in this case, *RandomForest* appears in knowledge base as the best algorithm in the 25% of the cases, which is a high percentage over 12 possible classifiers. For transversalsDS data set, it also recommends *RandomForest*, which is the 4th better classifier for this data set over 12.

Table 6. Recommender results

Data set	J48 Recommendation	NB Recommendation
multGlobalActivity	Jrip	RandomForest
mult2class2010	Jrip	RandomForest
transversalDS	J48	Jrip

These results demonstrate that our proposal is feasible although it is necessary to have a higher number of experiments in order to get a more general model. It is a little problem in e-learning context because although there are lots of courses hosted in e-learning platforms, not all courses can be used since it is necessary to know how the courses were designed and exploded by learners to be considered to predict performance.

We used other techniques based on landmarking [20] but the results were worse. On the other hand, we should add other meta-features related to parameter-setting of the algorithms. In this experimentation the algorithms were run with their default parameters.

5 Conclusions and future work

The application of data mining techniques are commonly known as a hard process generally based on trial and error empirical methods. As a consequence they can

only be applied by a small minority of experts. In this paper, a knowledge base is defined that contains information of previous data mining experiments in order to provide guidance to non-expert users to apply data mining techniques. To generate our knowledge base, a model-driven approach is defined, based on a Taverna workflow. As shown in our experiments, our knowledge base can be useful as a resource for non-expert data miners. The best classifiers can be recommended most of times from a set of 4 classifiers (*NaiveBayes*, *J48*, *Jrip*, and *BayesNet*) in order to predict the performance of students in our e-learning scenario. Moreover, in one of these cases, our knowledge base supports in recommending the best algorithm for two of the data sets. Although, the number of good recommendations were worse when the set of classifiers is 12, these results encourage us to continue researching in order to improve how the recommender can use the knowledge base in a better manner. As future work, we plan to conduct more experiments in order to study how to obtain better results when more classifiers are considered. Regardless the recommender can provide good results to a non-expert user with significantly low effort, more complex recommenders that improve these results could be developed.

Acknowledgments. This work has been partially funded by IN.MIND project from University of Alicante (Spain).

References

1. Bézivin, J.: On the unification power of models. Software and System Modeling 4(2), 171–188 (2005)
2. Blockeel, H., Vanschoren, J.: Experiment databases: Towards an improved experimental methodology in machine learning. In: Kok, J., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenic, D., Skowron, A. (eds.) Knowledge Discovery in Databases: PKDD 2007, Lecture Notes in Computer Science, vol. 4702, pp. 6–17. Springer Berlin / Heidelberg (2007), http://dx.doi.org/10.1007/978-3-540-74976-9_5, 10.1007/978-3-540-74976-9_5
3. Diamantini, C., Potena, D., Storti, E.: Ontology-driven kdd process composition. In: IDA. pp. 285–296 (2009)
4. Espinosa, R., Zubcoff, J.J., Mazón, J.N.: A set of experiments to consider data quality criteria in classification techniques for data mining. In: ICCSA (2). pp. 680–694 (2011)
5. Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P.: The kdd process for extracting useful knowledge from volumes of data. Commun. ACM 39(11), 27–34 (1996)
6. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: an update. SIGKDD Explorations 11(1), 10–18 (2009)
7. Hämäläinen, W., Vinni, M.: Comparison of machine learning methods for intelligent tutoring systems. In: Ikeda, M., Ashley, K., Chan, T.W. (eds.) Intelligent Tutoring Systems. Lecture Notes in Computer Science, vol. 4053, pp. 525–534. Springer Berlin / Heidelberg (2006), 10.1007/11774303_52
8. Hilario, M.: e-lico annual report 2010. Tech. rep., Université de Geneve (2010)
9. Hilario, M., Kalousis, A., Nguyen, P., Woznica, A.: A data mining ontology for algorithm selection and meta-mining. In: ECML/PKDD09 Workshop on Third Generation Data Mining: Towards Service-Oriented Knowledge Discovery. pp. 76–87. SoKD-09 (2009)
10. Hilario, M., Nguyen, P., Do, H., Woznica, A., Kalousis, A.: Ontology-based meta-mining of knowledge discovery workflows. In: Meta-Learning in Computational Intelligence, pp. 273–315 (2011)
11. Kalousis, A., Hilario, M.: Model selection via meta-learning: a comparative study. In: Tools with Artificial Intelligence, 2000. ICTAI 2000. Proceedings. 12th IEEE International Conference on. pp. 406–413 (2000)

12. Kietz, J.U., Serban, F., Bernstein, A., Fischer, S.: Designing kdd-workflows via htn-planning. In: Raedt, L.D., Bessière, C., Dubois, D., Doherty, P., Frasconi, P., Heintz, F., Lucas, P.J.F. (eds.) ECAI. Frontiers in Artificial Intelligence and Applications, vol. 242, pp. 1011–1012. IOS Press (2012)
13. Kleppe, A., Warmer, J., Bast, W.: MDA Explained. The Practice and Promise of The Model Driven Architecture. Addison Wesley (2003)
14. Kriegel, H.P., Borgwardt, K.M., Kröger, P., Pryakhin, A., Schubert, M., Zimek, A.: Future trends in data mining. Data Min. Knowl. Discov. 15(1), 87–97 (2007)
15. Mazón, J.N., Lechtenbörger, J., Trujillo, J.: A survey on summarizability issues in multidimensional modeling. Data Knowl. Eng. 68(12), 1452–1469 (2009)
16. Mazón, J.N., Zubcoff, J.J., Garrigós, I., Espinosa, R., Rodríguez, R.: Open business intelligence: on the importance of data quality awareness in user-friendly data mining. In: EDBT/ICDT Workshops. pp. 144–147 (2012)
17. Nisbet, R., Elder, J., Miner, G.: Handbook of Statistical Analysis and Data Mining Applications. Academic Press (2009)
18. Panov, P., Soldatova, L.N., Dzeroski, S.: Towards an ontology of data mining investigations. In: Discovery Science. pp. 257–271 (2009)
19. Parreiras, F.S., Staab, S., Winter, A.: On marrying ontological and metamodeling technical spaces. In: Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering. pp. 439–448. ESEC-FSE '07, ACM, New York, NY, USA (2007), http://doi.acm.org/10.1145/1287624.1287687
20. Pfahringer, B., Bensusan, H., Giraud-carrier, C.: Meta-learning by landmarking various learning algorithms. In: Proceedings of the 17th International Conference on Machine Learning. pp. 743–750 (2000)
21. Romero, C., Ventura, S.: Educational Data Mining: A Review of the State-of-the-Art. IEEE Tansactions on Systems, Man and Cybernetics, part C: Applications and Reviews 40(6), 601–618 (2010)
22. Soldatova, L., King, R.: An ontology of scientific experiments. J R Soc Interface 3(11), 795–803 (2006)
23. Vanschoren, J., Blockeel, H.: Stand on the Shoulders of Giants: Towards a Portal for Collaborative Experimentation in Data Mining. International Workshop on Third Generation Data Mining at ECML PKDD 1, 88–89 (Sep 2009)
24. Vanschoren, J., Blockeel, H., Pfahringer, B., Holmes, G.: Experiment databases - a new way to share, organize and learn from experiments. Machine Learning 87(2), 127–158 (2012)
25. Vanschoren, J., Soldatova, L.: Exposé: An ontology for data mining experiments. In: International Workshop on Third Generation Data Mining: Towards Service-oriented Knowledge Discovery (SoKD-2010),. pp. 31–46 (Sep 2010)
26. Vilalta, R., Giraud-Carrier, C.G., Brazdil, P., Soares, C.: Using meta-learning to support data mining. IJCSA 1(1), 31–45 (2004)
27. Wu, X., Kumar, V., Ross Quinlan, J., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G.J., Ng, A., Liu, B., Yu, P.S., Zhou, Z.H., Steinbach, M., Hand, D.J., Steinberg, D.: Top 10 algorithms in data mining. Knowl. Inf. Syst. 14(1), 1–37 (Dec 2007), http://dx.doi.org/10.1007/s10115-007-0114-2
28. Záková, M., Kremen, P., Zelezný, F., Lavrac, N.: Automating knowledge discovery workflow composition through ontology-based planning. IEEE T. Automation Science and Engineering 8(2), 253–264 (2011)
29. Zorrilla, M.E., García-Saiz, D.: Business Intelligence Applications and the Web: Models, Systems and Technologies, chap. Mining Service to Assist Instructors involved in Virtual Education. Information Science Reference (IGI Global Publishers) (September 2011)

Using Semantic Lifting for improving Process Mining: a Data Loss Prevention System case study

Antonia Azzini, Chiara Braghin, Ernesto Damiani, Francesco Zavatarelli

Dipartimento di Informatica
Università degli Studi di Milano, Italy
{antonia.azzini, chiara.braghin, ernesto.damiani,
francesco.zavatarelli}@unimi.it

Abstract. Process mining is a process management technique to extract knowledge from the event logs recorded by an information system. We show how applying an appropriate semantic lifting to the event and workflow log may help to discover the process that is actually being executed. In particular, we show how it is possible to extract not only knowledge about the structure of the process, but also to verify if some non-functional properties, such as security properties, hold during the process execution.

1 Introduction

Business Process Intelligence (BPI) is a research area that is quickly gaining interest and importance: it refers to the application of various measurement and analysis techniques both at design and at run-time in the area of business process management. In practice, BPI stands for an integrated set of tools for managing process execution quality by offering several features such as monitoring, analysis, discovery, control, optimization and prediction.

In particular, *process mining* is a process management technique to extract knowledge from the event logs recorded by an information system. It is often used for discovering processes if there is no a priori model, or for conformance analysis in case there is an a priori model that is compared with the event log in order to find out if there are discrepancies between the log and the model.

In this work, we focus our attention on process mining techniques based on the computation of frequencies among event dependencies to reconstruct the workflow of concurrent systems. In particular, we show how applying an appropriate *semantic lifting* to the event and workflow log may help to discover the process that is actually being executed. In the Web scenario, the term semantic lifting refers to the process of associating content items with suitable semantic objects as metadata to turn unstructured content items into semantic knowledge resources. In our case, the semantic lifting procedure corresponds to all the

transformations of low-level systems logs carried out in order to achieve a conceptual description of business process instances, without knowing the business process a priori.

To illustrate our proposal, we present a case study based on a *data loss prevention scenario* aiming to preventing the loss of critical information in companies. In order to describe our running example, we use a lightweight data representation model designed to support real time monitoring of business processes based on a shared vocabulary defined using open standard representations (RDF). We believe that the usage of RDF as modeling language allows independence and extremely flexible interoperability between applications.

The contributions of this paper are:

- an example on how semantic lifting may help to improve the discovering process during process mining;
- a definition of a Data Loss Prevention System in RDF, modeling a multi-level security policy based on the organizational boundaries (internal vs external actors and resources);
- an example on how, using semantic lifting in combination with standard process mining techniques during the discovery phase, it is possible to extract not only knowledge about the structure of the process, but also to verify if some non-functional properties, such as security properties, hold during the process execution.

This work is organized as follows. In Section 2, we introduce the semantic lifting approach and describe how it has been used so far; in Section 3, we give a short overview of the Resource Description Framework (RDF). Section 4 is the core of the paper, where we present the Data Loss scenario and we give some examples on how semantic lifting helps improving the investigation on the process. Section 5 concludes the paper.

2 Semantic Lifting: State of the Art

In the Web scenario, the term semantic lifting refers to the process of associating content items with suitable semantic objects as metadata to turn unstructured content items into semantic knowledge resources. As discussed in [3], by semantic lifting we refer to all the transformations of low-level systems logs carried out in order to achieve a conceptual description of business process instances. Typically, this procedure is implicitly done by converting data from the data storages of an information system to an event log format suitable for process monitoring [5]. We believe that this problem is orthogonal to the abstraction problem in process mining, dealing with different levels of abstraction when comparing events with modeled business activities [4]: our goal is to see how associating some semantics to an event from the log it is possible to extract better knowledge about some properties of the overall process, not to see which is the mapping between events and business activities/tasks.

So far, the term semantic lifting has been used in the context of model-driven software development. In [9], the authors proposed a technique for designing and implementing tool components which can semantically lift model differences arising among the tools. In particular, they used the term semantic lifting of differences to refer to the transformation of low-level changes to all the more conceptual descriptions of model modifications.

The literature [11] reports how the Business Process Management (BPM) usually operates at two main distinct levels, corresponding, respectively, to a management level, supporting business organizations in optimizing their operational processes, and a technology level, supporting IT users in process modeling and execution.

In these two levels, experts operate without a systematic interaction and cooperation, causing the well known problem of Business/IT alignment. In fact, one key problem is the alignment of different tools and methods used by the two communities (business and IT experts). In order to reduce the gap between these two levels, De Nicola and colleagues refer in [11] to *s*emantic technologies as an useful approach at supporting business process design, reengineering and maintenance of the business process, by highlighting some advantages related to the semantic lifting. The first one regards the support that the semantic lifting can give to business process design by a semantic alignment of a business process respect to a reference ontology. The semantic alignment can be achieved by performing consistency checking through the use of a reasoning engine. Then, the reengineering of a business process (BP) can be improved by providing suggestions to experts during the design phase of a BP, for example in finding alternative elements with semantic search and similarity reasoning over the business ontology. The authors also indicate, as another advantage, the possibility to support a BP maintenance by automatically checking the alignment between one of more business processes against the business ontology when the latter is modified. This can provide strong benefits since, for instance, a change in the company organization, could affect many business processes that need to be manually checked.

3 RDF to model Business Processes

Generally speaking, the Resource Description Framework (RDF) [7] corresponds to a standard vocabulary definition, which is at the basis of the Semantic Web vision, and it is composed by three elements: concepts, relations between concepts and attributes of concepts. These elements are modeled as a labelled oriented graph [6], defined by a set of triples $< s, p, o >$ where s is subject, p is predicate and o is object, combined as shown in Figure 1.

New information is inserted into an RDF graph by adding new triples to the set. It is therefore easy to understand why such a representation can provide big benefits for real time business process analysis: data can be appended 'on the fly' to the existing one, and it will become part of the graph, available for any

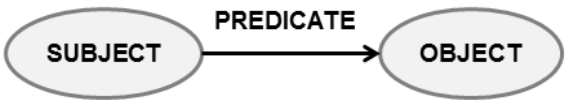


Fig. 1. RDF subject-object relation.

analytical application, without the need for reconfiguration or any other data preparation steps.

RDF standard vocabularies allow external applications to query data through SPARQL query language [12]. SPARQL is a standard query language for RDF graphs based on conjunctive queries on triple patterns, identifying paths in the RDF graph. Thus, queries can be seen as graph views. SPARQL is supported by most of the triples stores available.

Moreover, RDF provides a basic set of semantics that is used to define concepts, sub-concepts, relations, attributes, and can be extended easily with any domain-specific information. For this reason, it is an extremely generic data representation model that can be used in any domain.

In [10], the authors present a framework based on RDF for business process monitoring and analysis. They define an RDF model to represent a generic business process that can be easily extended in order to describe any specific business process by only extending the RDF vocabulary and adding new triples to the triple store. The model is used as a reference by both monitoring applications (i.e., applications producing the data to be analyzed) and analyzing tools. On one side, a process monitor creates and maintains the extension of the generic business process vocabulary either at start time, if the process is known a priori, or at runtime while capturing process execution data, if the process is not known. Process execution data is then saved as triples with respect to the extended model. On the other side, the analyzing tools may send SPARQL queries to the continuously updated process execution RDF graph.

Figure 2 shows the conceptual model of a generic business process, seen as a sequence of different tasks, each having a start/end time and possibly having zero or more sub-tasks. We will use this model to describe our running example.

4 Case study: a Data Loss Prevention System

Data loss is an error condition in information systems in which information is destroyed by failures or neglect in storage, transmission, or processing. Consider for example some different companies belonging to the same manufacturing supply chain and sharing business process critical data by using a file sharing server in order to access to the data. This scenario could expose the critical data to malicious users if access control is not implemented correctly. Indeed, an access control to such a server should be carried out by a security model, based on specific rules considering, for example, the user authentication for file sharing,

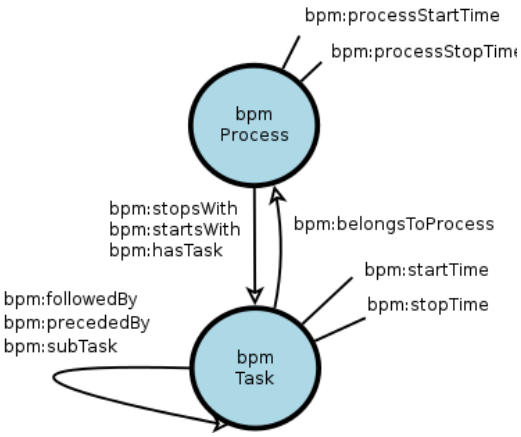


Fig. 2. RDF Representation of a generic business process.

security policies definition for users that have access rights to some confidential data, data checking before sending them to external companies that do not belong to the manufacturing supply chain considered, usage of authorized channels for data delivery, like company's e-mail, and so on. In order to prevent any kind of data loss, systems usually develop an intellectual ownership defense, also called *data-loss model*, tracking any action operated on a document. The process is able to highlight some security-related information, such as the economical value assigned to the outgoing intellectual ownership, or the number of 'confidential' data sent around, possibly to external destinations.

Protecting the confidentiality of information stored in a computer system, or transmitted over a public network is a relevant problem in computer security, called *information leakage*. The approach of information flow analysis involves performing a static analysis of the program with the aim of proving that there will not be leaks of sensitive information. The starting point in secure information flow analysis is the classification of program variables into different security levels (i.e., defining a multi-level security policy). In the simplest case, two levels are used: public (or low, L) and secret (or high, H). There is an information flow from object x to object y whenever the information stored in x is transferred to, or used to derive information transferred to, object y. The main purpose is to prevent leak of sensitive information from an high variable to a lower one.

In our case study, we will consider the two security levels generated by the organizational boundaries (internal/external).

4.1 An RDF Model of the Data Loss case study

Our RDF model representation of the Data Loss case study is based on the lightweight RDF data model for business processes analysis carried out in [10] and briefly described in Section 3. As previously pointed out, the model does not contain any data, but it only provides a generic schema that the process

monitoring applications extend and instantiate. The Data Loss RDF model is then defined as an extension of the general schema and is depicted in Figure 3: it consists of the conceptual model of Figure 2 extended with domain specific concepts taken from the Data loss problem scenario. In particular, in our running example, we assume that the files shared between the companies in the same manufacturing chain are CAD files, thus the business process is named 'pCAD:CAD Process'.

Fig. 3. RDF Representation of the Data Loss Problem.

The main task is 'pCAD:UseFile', which creates a data file ('pCAD:File'), that can be used by a project manager ('pCAD:Employee'), which is an employee of one of the companies. All the concepts labeled 'dLoss' are the one which specify the security model to prevent Data Loss. The subclass called 'dLOSS:File' is assigned several attributes: the security level ('dLoss:ConfidentialFile'), and the economic value ('dLOSS:economicValue'), in order to be able to check information leakage or to compute the economical loss of the outgoing intellectual ownership. The destination of an operation on a file ('dLOSS:Destination') has three main attributes: 'dLOSS:InternalNetwork', 'dLOSS:ValueNetworkActor' and 'dLOSS:ExternalActor', the last specifying if the file has been shared with a user within the organizational boundaries or not.

As reported in Section 3, the usage of a framework based on an RDF triple store like [10] is specifically designed for integrating multiple source and supporting fast and continuous execution of SPARQL queries favouring the join between the execution processes and the monitoring.

4.2 Semantic Lifting for Mining a Data Loss Process

In this section, we show how an appropriate semantic lifting may help during the process mining phase. Process mining is the technique of distilling a struc-

tured process description from a set of real executions. To the sake of discussion, we limit our example to process mining algorithms that are based on detecting ordering relations among events to characterize a workflow execution log [14]. In particular, they build dependency/frequencies tables that are used to compare single executions in order to induce a reference model, or to verify the satisfiability of specific conditions on the order of executions of events. We assume that the reader is familiar with the following definitions that are common in this scenario [2].

Workflow trace. Let $E = \{e_1, e_2, \cdots, e_n\}$ be a set of events, then $t \in E^*$ is a *workflow (execution) trace.*

Workflow log. Let $E = \{e_1, e_2, \cdots, e_n\}$ be a set of events, then $W \subseteq E^*$ is a *workflow log.*

Successor. Let W be a workflow log over E and $a, b \in E$ be two events, then b is a *successor* of a (notation $a \prec_W b$) if and only if there is a trace $t \in W$ such that $t = \{e_1, e_2, \cdots, e_n\}$ with $e_i \equiv a$ and $e_{i+1} \equiv b$. Similarly, we use the notation $a \prec_W^n b$ to express that event b is *successor* of event a **by** n **steps** (i.e., $e_i \equiv a$ and $e_{i+k} \equiv b$, with $1 < k \leq n$).

Notice that the successor relationship is rich enough to reveal many workflow properties since we can construct dependency/frequency tables that allow to verify the relations that constraint a set of log traces. However, in order to better characterize the significance of dependency between events, other measures, based on information theory, are adopted in the literature, such as for instance the J-Measure proposed by Smyth and Goodman [13], able to quantify the information content of a rule.

Table 1 shows a fragment of a workflow log possibly generated by a data loss prevention system tracking in-use actions based on the RDF model described in the previous section. The system reports all the events that generated a new status of a specific document. In particular, we assume that for each event it is specified: (*i*) the type of event (Create, Update, Share, Remove); (*ii*) the user performing the action on the file expressed by the email address; (*iii*) the timestamp spotlighting the end point (a system user, in our case) that achieved the control on the document at the end of the event which allow us to chronologically order the events; and (*iv*) the estimated value of the file (in the range: Low, Medium, High).

Following the approach in [14], we construct the dependency/frequency (D/F) table from the data log illustrated in Table 1. More in detail, the information contained in Table 2, are:

- the overall frequency of event a (notation $\#a$);
- the frequency of event a followed by event Create (C for short);
- the frequency of event a followed by event Update (U for short) by 1, 2 and 3 steps;
- the frequency of event a followed by event Share (S for short) by 1, 2 and 3 steps.

Table 1. An example of workflow log for the Data Loss case study.

Event	User	Timestamp	Status	Estimated value
File AAAA				
Create	userP@staff.org	2012-11-09 T 11:20	Draft	High
Update	userP@staff.org	2012-11-09 T 19:20	Draft	
Share	userA@staff.org	2012-11-12 T 10:23	Proposal	
Update	userA@staff.org	2012-11-14 T 18:47	Proposal	
Share	userP@staff.org	2012-11-15 T 12:07	Proposal	
Update	userP@staff.org	2012-11-18 T 09:21	Recommendation	
Share	userM@inc.org	2012-11-18 T 14:31	Recommendation	
File AAAB				
Create	userF@staff.org	2012-12-03 T 09:22	Draft	Medium
Update	userF@staff.org	2012-12-03 T 12:02	Draft	
Update	userF@staff.org	2012-12-03 T 17:34	Draft	
Share	userV@staff.org	2012-12-05 T 11:41	Draft	
Share	userD@staff.org	2012-12-05 T 11:41	Proposal	
Update	userD@staff.org	2012-12-08 T 10:36	Proposal	
Update	userV@staff.org	2012-12-08 T 16:29	Proposal	
Share	userG@inc.org	2012-12-10 T 08:09	Proposal	
Update	userV@staff.org	2012-12-10 T 18:38	Recommendation	
File AAAC				
Create	userV@staff.org	2012-12-04 T 10:26	Draft	Medium
Update	userV@staff.org	2012-12-04 T 13:12	Draft	
Update	userV@staff.org	2012-12-05 T 10:12	Draft	
Share	userA@staff.org	2012-12-05 T 12:22	Draft	
Share	userD@staff.org	2012-12-06 T 14:51	Proposal	
Share	userM@inc.org	2012-12-07 T 10:31	Proposal	

Using this table we can observe that the following patterns hold in W: $Create \succ_W Update$ or $Create \succ_W^* Share$, that is, a file is always created before being updated or shared.

Table 2. An example of Dependency/Frequency based on the Successor relation.

a	$\#a$	$a \prec$ C	$a \prec$ U	$a \prec^2$ U	$a \prec^3$ U	$a \prec$ S	$a \prec^2$ S	$a \prec^3$ S
Create	3	0	3	2	1	0	1	2
Update	10	0	3	3	2	6	5	5
Share	9	0	4	2	2	3	3	1

Since a 'data-loss model' is typically aimed at detecting anomalous behaviors, the expected behavior in the form of unwanted behaviors (black-listing) or wanted behavior (white-listing) needs to be defined. This can be done by identifying behavioral patterns over the sequences of events that are normally registered in the workflow logs. We may, for instance, be interested in mining expected behavior for documents shared within and outside the boundaries of the organization. Still focusing our attention on the Share events which might cause unwanted information flows, we might be interested to see which are the users that most frequently share the documents with other users either inside or outside the boundaries. To this aim, a semantic lifting procedure can be applied to the log data for remodeling the representation of the process and allowing additional investigations.

A first semantic lifting can be done by applying the Data Loss model described in the previous section to our log, in order to distinguish among events where files are shared internally or externally to the organization. In our example, the lifting can be done by exploiting two data transformations rules expressed according to Equation 1. Data are then mapped to the model using standard techniques for mapping RDF data [8].

$$\begin{gathered} User \| [A-Z0-9\_.-] + @staff + . [A-Z] \{2,4\} \\ \rightarrow dLOSS : Internal \\ [A-Z0-9\_.-] + @inc + . [A-Z] \{2,4\} \\ \rightarrow dLOSS : External \end{gathered} \tag{1}$$

After applying the semantic lifting to the log, we are able to build and fill Table 3, where a Share event is rewritten as 'Share Internal' when the Share event is performed by an Internal user, otherwise the event is rewritten as a 'Share External' event. In this new dependences/frequencies among events, we observe that a new pattern holds: $ShareInternal \succ_W ShareInternal \succ_W ShareExternal$. Informally, we can interpret this pattern in the execution traces as the identification of an expected behavior about document sharing: before a document is shared externally to the organization it has to pass some (typically two) internal steps.

Table 3. D/F after a first semantic lifting: Sharing between Internal/External Users.

a	#	$a \prec$ SI	$a \prec^2$ SI	$a \prec$ SE	$a \prec^2$ SE
Share Internal (SI)	6	3	0	3	3
Share External (SE)	3	0	0	0	0
SI $\prec$ SI	3	0	0	0	3

Another semantic lifting can be done by grouping together all the Share log events performed by the same user. Please notice that, as described in detail in Section 3, RDF allows to easily aggregate data by considering their shared properties. Moreover, SPARQL queries allow us to manipulate data to view them in the appropriate structural order, by defining, for example, events that are grouped and aggregated by different attributes. Table 4 reports the frequencies of these events, referring to an event `Share` with `userV@staff.org` as `SV`, `Share` with `userA@staff.org` as `SA`, and so on.

Table 4. D/F after another semantic lifting: Sharing events for specific Users.

a	#	$a \prec$ SA	$a \prec^2$ SA	$a \prec$ SD	$a \prec^2$ SD	$a \prec$ SG	$a \prec^2$ SG	$a \prec$ SM	$a \prec^2$ SM	$a \prec$ SP	$a \prec^2$ SP	$a \prec$ SV	$a \prec^2$ SV
SA	2	0	0	1	0	0	0	0	1	0	1	0	0
SD	2	0	0	0	0	0	0	1	0	0	0	0	0
SG	1	0	0	0	0	0	0	0	0	0	0	0	0
SM	2	0	0	0	0	0	0	0	0	0	0	0	0
SP	1	0	0	0	0	0	0	0	1	0	0	0	0
SV	1	0	0	1	0	0	0	0	0	0	0	0	0

We can observe that the table is sparse, therefore few patterns can be proved to hold in W. In our example, for instance, we can derive that in only one case $SA \succ^2_W SM$, meaning that User `userA@staff.org` shares a document before the same document is shared by User `userM@staff.org`. We can also derive that User `userM@staff.org` is always the last to share the document, possibly meaning that he is at the bottom of the organization hierarchy or that he is an untrusted user (thing that is supported by the fact that it is an external user). Given the low frequency of both cases, the two conclusions we drew are not particularly relevant since they are not supported by a large number of traces. The sparsity of the table is typical of so called 'spaghetti-like processes', i.e., unstructured processes where recurrent event sequences are not so easily defined [1]. In this case, a semantic lifting procedure could be applied to the log data for remodeling the representation of the process and implementing additional investigations.

5 Conclusion

In this paper we showed how standard process mining techniques can be combined with semantic lifting procedures on the workflow logs in order to discover more precise workflow models from event-based data. Moreover, we highlighted the benefits using RDF as a modeling formalism by using it in our case study. This is just a first step to show the feasibility and the advantages of the approach. As a future work we plan to study how to automatize the process by exploiting the usage of RDF as a modeling language.

Acknowledgment

This work was partly funded by the Italian Ministry of Economic Development under the Industria 2015 contract - `KITE.IT` project.

References

1. Van der Aalst, W.M.P.: Process mining: Discovering and improving spaghetti and lasagna processes. Keynote Lecture, IEEE Symposium Series on Computational Intelligence (SSCI 2011)/IEEE Symposium on Computational Intelligence and Data Mining (CIDM 2011) (April 2011)
2. Van der Aalst, W.M.P., van Dongen, B.F., Herbst, J., Maruster, L., Schimm, G., Weijters, A.J.M.M.: Workflow mining: a survey of issues and approaches. Data Knowl. Eng. 47(2), 237–267 (Nov 2003), http://dx.doi.org/10.1016/S0169-023X(03)00066-1
3. Azzini, A., Ceravolo, P.: Consistent process mining over big data triple stores. In: Proceedings of the IEEE International Conference on Big Data. p. to appear. IEEE Publisher, June 27-July 2, 2013, Santa Clara Marriott, CA, USA (2013)
4. Baier, T., Mendling, J.: Bridging abstraction layers in process mining by automated matching of events and activities. In: Daniel, F., Wang, J., Weber, B. (eds.) Business Process Management, Lecture Notes in Computer Science, vol. 8094, pp. 17–32. Springer Berlin Heidelberg (2013)
5. Buijs, J.: Mapping data sources to xes in a generic way, master's thesis (2010)
6. Carroll, J., Bizer, C., Hayes, P., Stickler, P.: Named graphs. Journal of Web Semantics 3(3) (2005)
7. Hayes, P., McBride, B.: Resource description framework (rdf) (2004), http://www.w3.org/
8. Hert, M., Reif, G., Gall, H.C.: A comparison of rdb-to-rdf mapping languages. In: Proceedings of the 7th International Conference on Semantic Systems. pp. 25–32. I-Semantics '11, ACM, New York, NY, USA (2011), http://doi.acm.org/10.1145/2063518.2063522
9. Kehrer, T., Kelter, U., Taentzer, G.: A rule-based approach to the semantic lifting of model differences in the context of model versioning. In: Automated Software Engineering (ASE), 2011 26th IEEE/ACM International Conference on. pp. 163–172 (2011)
10. Leida, M., Majeed, B., Colombo, M., Chu, A.: Lightweight rdf data model for business processes analysis. Data-Driven Process Discovery and Analysis, Series: Lecture Notes in Business Information Processing 116 (2012)

11. Nicola, A.D., Mascio, T.D., Lezoche, M., Tagliano, F.: Semantic lifting of business process models. 2012 IEEE 16th International Enterprise Distributed Object Computing Conference Workshops 0, 120–126 (2008)
12. Prudhommeaux, E., Seaborne, A.: Sparql query language for rdf (2008), http://www.w3.org/
13. Smyth, P., Goodman, R.M.: Rule induction using information theory. Knowledge discovery in databases 1991 (1991)
14. Van Der Aalst, W., Van Hee, K.: Workflow management: models, methods, and systems. MIT press (2004)

Challenges of Applying Adaptive Processes to Enable Variability in Sustainability Data Collection

Gregor Grambow, Nicolas Mundbrod, Vivian Steller and Manfred Reichert

Institute of Databases and Information Systems
Ulm University, Germany
{gregor.grambow,nicolas.mundbrod,vivian.steller,manfred.reichert}@uni-ulm.de
http://www.uni-ulm.de/dbis

Abstract. Nowadays, demanding legal regulations as well as sophisticated customer needs force companies in electronics and automotive industries to provide a multitude of different sustainability indicators. Since their products usually contain numerous components and sub-components, companies must deal with complex, intransparent data collection processes along their supply chains in order to finally deliver valuable data. A myriad of different automatic and manual tasks, potentially long-running processes, and quickly changing situations result in great variability that is hard to handle. In the SustainHub project, a dedicated information system for supporting data collection processes is developed. Thereby, core challenges as well as state-of-the-art were systematically gathered, consolidated as well as assessed. The condensed results are presented in this paper.

Key words: Business Process Variability, Data Collection, Sustainability, Supply Chain

1 Introduction

These days, companies of the electronics and automotive industry face steadily growing demands for sustainability compliance triggered by authorities, customers and public opinion. As products often consist of numerous individual components, which, in turn, also comprise sub-components, heterogeneous sustainability data need to be collected along intertwined and intransparent supply chains. Thereby, highly complex, cross-organizational data collection processes are required, featuring a high variability, e.g., through dynamically integrating companies' employees and information systems (ISs). Further issues include incompleteness and varying quality of provided data, heterogeneity of data formats, or changing situations and requirements. Until today, there is no dedicated IS supporting companies in creating, managing and optimizing such data

collection processes. Within the SustainHub[1] project, such a dedicated information system is being developed. In this context, use cases, delivered by industry partners from the automotive and the electronics domain, have been intensively studied in order to consolidate core challenges and essential requirements regarding the IT-support of data collection processes. In relation, state-of-the-art has also been deeply studied to assess whether existing approaches and solutions satisfy the requirements. As a result, this paper systematically presents the condensed core challenges and state-of-the-art considering complex sustainability data collection process along today's supply chains. This domain is well suited for eliciting such challenges because of the complexity of the supply chains on the one hand and the requirements imposed by emerging laws and regulations on the other. However, they can be transferred to many other domains as well. Thus, this contribution identifies 7 core challenges for data exchange and collection in complex distributed environments and also reviews approaches in place to solve these challenges. Thereupon, future research in the area of adaptive business process management can be aligned to extend existing approaches for supporting more variability and dynamics in today's business processes.

Therefore, the fundamentals and an illustrating example are introduced in section 2. Subsequently, seven data collection challenges are unveiled in section 3, exposing concrete findings, identified problems and derived requirements. In section 4, the current state-of-the-art is presented based on its origin. Finally, section 5 rounds out this paper giving a conclusion and an outlook.

2 Sustainable Supply Chains

This section elaborates on the domain of sustainable supply chains and gives background information.

2.1 Fundamentals

In today's globalized industry, the development and production of many products is based on intransparent, complex supply chains with dozens of interconnected companies distributed around the globe. To ensure and extend competitiveness, complex communication tasks must be managed properly for effective and efficient interorganizational processes. Generally, such cross-organizational collaboration involves a variety of different manual and automated tasks. Involved companies significantly differ in size and industry background, and they use various different ISs, which are not able to intercommunicate easily. Due to this heterogeneity, neither federated data schemes, unifying tools nor other concepts can be realistically introduced without considerable effort [1].

As sustainability is is an emerging trend, companies even face a new challenge in their supply chains: sustainable development and production. The incentives

[1] SustainHub (Project No.283130) is a collaborative project within the 7th Framework Programme of the European Commission (Topic ENV.2011.3.1.9-1, Eco-innovation).

are given by two parties: On one hand, legal regulations, increasingly issued by authorities, force companies to publish more and more sustainability indicators (like greenhouse gas emissions in production or gender issues) on an obligatory basis. On the other hand, public opinion and customers compel companies to provide sustainability information (e.g., organic food) as an important base for their purchase decisions.

Examples include ISO 14000 standard for environmental factors in production, GRI[2] covering sustainability factors or regulations like REACH[3] and RoHS[4]. Overall, sustainability information involve a myriad of different indicators. It relates to social issues (e.g., employment conditions or gender issues), to environmental issues (e.g., hazardous substances or greenhouse gas (GHG) emissions), or to managerial issues (e.g., compliance issues).

There already exist tools at market providing support for the management and transfer of sustainability data: IMDS[5] (International Material Data System), for instance, is used in the automotive industry and allows for material declaration by creating and sharing bills of materials (BOM). A similar system exists for the electronics industry (Environ BOMcheck[6]). Despite providing useful support in basic data declaration and exchange tasks, these tools clearly fall short in providing dedicated support for the sustainability data collection and exchange along the supply chains.

2.2 Illustrating Example

To illustrate the complexity of sustainability data collection processes in a distributed supply chain, we provide an example. The latter was composed with the problems and requirements provided by SustainHub's partner companies for the automotive and electronics industry by formal and informal surveys and interviews. Please mind that data collection in such a complex environment does not have the characteristics of a simple query. It is rather a varying, long-running process incorporating various activities and involving different participants.

The example illustrated in Fig. 1, depicts the following situation: Imposed by regulations, an automotive manufacturer (requester) has to provide sustainability data considering its production. This data is captured by two sustainability indicators, one dealing with the greenhouse gas emissions relating to the production of a certain product, the other addressing the REACH regulation. The latter concerns the whole company as companies usually declare compliance to that regulation on a company basis.

To provide data regarding these two indicators, the manufacturer has to gather related information from his suppliers (answerer). Hence, it requests a

[2] Global Reporting Initiative: `https://www.globalreporting.org`

[3] Regulation (EC) No 1907/2006: Registration, Evaluation, Authorisation and Restriction of Chemicals

[4] Directive 2002/95/EC: Restriction of (the use of certain) Hazardous Substances

[5] `http://www.mdsystem.com`

[6] `https://www.bomcheck.net`

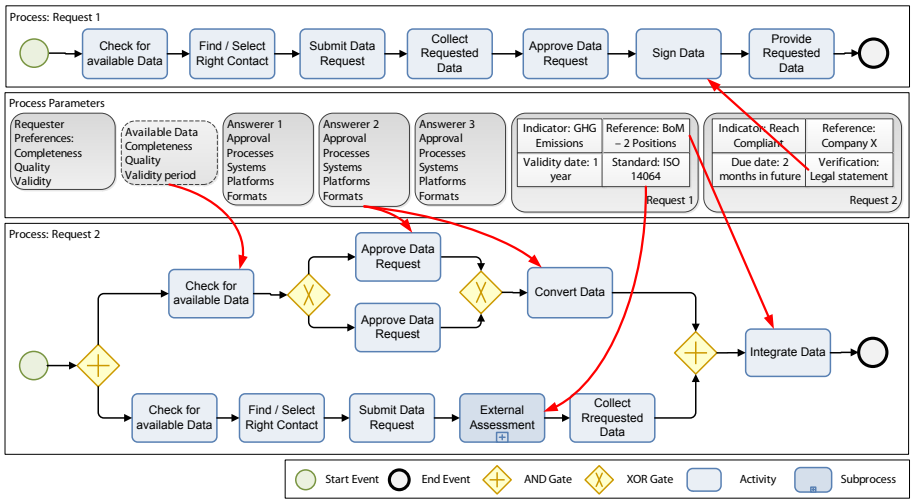


Fig. 1: Examples of Two Data Collection Processes

REACH compliance statement from one of its suppliers. To get the information, the activities shown in the process *Request 1* have to be executed. Furthermore, the product for which the greenhouse gas emissions shall be indicated has a BoM with two positions coming from external suppliers. Thus, the request, depicted by the second workflow, has to be split up into two requests, one for each supplier.

Hence, the basic scenario involves a set of activities as part of the data collection processes. Some of these are common for the requests, e.g., on the requester side, checking available data that might satisfy the request, selecting the company and contact person, and the submitting the request. On the answerer side, data must be collected and provided. The other process activities are specifically selected for each case. Thereby, the selection of the right activities is strongly driven by data (process parameters) coming from the requester, the answerer, the requests and indicators, and possible already available data.

For example, *Request 1* implies a legally binding statement considering REACH compliance. Therefore, a designated representative (e.g., the CEO) must sign the data. In many cases, companies have special authorization procedures for releasing of such data, e.g., that one or more responsible persons have to approve the request (cf. two parallel approval activities (*Approve Data Request*) at *Request 2*, *four-eyes-principle*). In some cases, data may be already available in a company and does not have to be manually gathered (cf. *Request 2*, *Check of available Data*). However, every time the company-internal format of the answerer does not match the requester's one, a conversion must be applied. Further, some indicators and requests also directly relate to a given standard (e.g., ISO 14064 for greenhouse gases) where this can directly trigger an assessment of the answerer if he cannot exhibit the fulfillment of the standard (cf. *Request 2*, *External Assessment*).

Finally, another important aspect for often long-running data collection processes is that process parameters might change over time and, hence, exceptional situations could occur. Even in this very simple example, many variations and deviations might occur: for example, if the CEO was not available, activity *Sign Data* could be delayed. In turn, this might become a problem if there are defined deadlines for the query answer.

3 Data Collection Challenges

Following first insights provided in Section 2, this section presents seven concrete challenges for an information system supporting sustainability data collection processes along a supply chain (IS-DCP). The results are based on findings from case studies conducted with industrial partners in the SustainHub project. Three figures serve for illustration purposes: Fig. 2 illustrates data collection challenges (DCC) 1 and 2, Fig. 3 illustrates DCC 3 and 4, and Fig. 4 illustrates DCC 5-7.

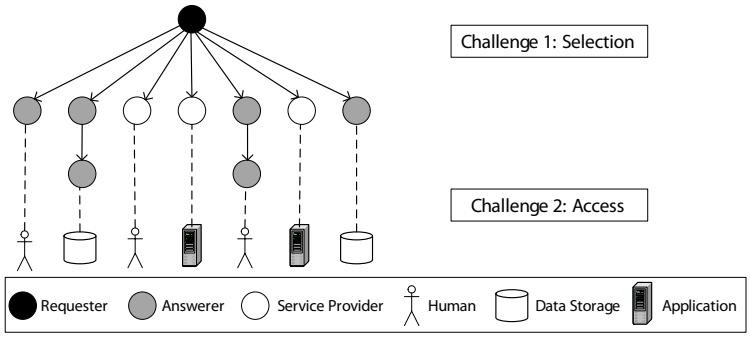


Fig. 2: Data Collection Challenges 1 and 2

3.1 DCC 1: Dynamic Selection of Involved Parties

Findings Sustainability data collection in a supply chain involves various parties. A single request may depend on the timely delivery of data from different companies. For manual tasks, this mostly has to be done by a specific person with sustainability knowledge or authority. In big companies, it can be even difficult to find the right contact person to answer a specific request. In relation, contact persons may change from time to time. Furthermore, as the requested data is often complex, has to be computed, or relates to legal requirements, external service providers may be involved in the data collection request as well. Finally, regarding the timely answering of a request, many requests are adjusted and forwarded to further suppliers (cf. Fig. 2) – thus answering times can multiply.

Problems The contemporary approach to such requests heavily relies on individuals conducting manual tasks and interacting individually. There are tools (e.g., email) which can provide support for some of these and partly automate them. However, much work is still coordinated manually. As a request can be forwarded down the supply chain, it is quite difficult to predict, who exactly will be involved in its processing. Resulting from that, answering times of requests can be hardly estimated in a reliable manner as well.
Requirements An IS-DCP need to enable companies to centrally create and manage data collection requests. Thereby, it must be possible to simplify the dynamic selection process of involved parties and contact persons regarding the request answerers as well as potentially needed service providers. This is a basic requirement for enabling efficient request answering, data management, and monitoring.

3.2 DCC 2: Access to Requested Data

Findings In a supply chain different parties follow different approaches to data management. Big companies mostly have implemented a higher level of automation while SMEs heavily rely on the work of individual persons. Furthermore, sustainability reporting is a relatively new area and a unified reporting method is not implemented along supply chains. This implies great variability when it comes to accessing companies' internal data. Some companies have advanced software solutions for their data management, some manage their data in generic databases, some store it in specific files (e.g., Excel), and some have even not started to manage sustainability data yet.
Problems The contemporary approach to sustainability reporting is managed manually to a large extend. This involves manual requests from one party to another and different data collection tasks on the answerer side. This can impose large delays in data collection processes as sustainability data must be manually gathered from systems, databases or specific files before it can be compiled, prepared and authorized in preparation to the delivery to the requester.
Requirements An IS-DCP must accelerate and facilitate the access to requested sustainability data. On the one hand, this includes guiding users in manual data collection as well as automizing data-related activities (e.g., data approval, data transformation) as far as possible. On the other hand, automatic data collection should be enabled whenever possible. This involves accessing the systems containing the data automatically (e.g., via the provision of appropriate interfaces) and including such activities with manual approval activities when needed. Finally, data conversion between different formats ought to be supported as a basis for data aggregation.

3.3 DCC 3: Meta Data Management

Findings The management and configuration of sustainability data requests in a supply chain relies on a myriad of different data sets. As aforementioned,

this data comes from various sources. Examples of such parameters include the preferences of the requester as well as the answerers (including approval processes and data formats) or the properties of the sustainability indicators (e.g., relations to standards) (cf. Fig. 3). As a result, potentially matching data might be already available in some cases but exposing different properties as requested.
Problems As requests rely on heterogeneous data, they are difficult to manage. Requirements are partially presumed by the requester and often implicit. Hence, answerers might be unaware of all requirements and deliver data not matching them. Moreover, it is difficult to determine whether data, which has been collected before, fits the requirements of a new request. Finally, as a supply chain might involve a large number of requesters and answerers, this problem multiplies as crucial request data is scattered along the entire supply chain.
Requirements To be able to consistently and effectively manage data collection processes, an IS-DCP must centrally implement, manage and provide an understandable meta data schema addressing relevant request parameters. Thereby, instanced data based on the uniform meta data schema can be effectively used to directly derive and adjust variants of data collection processes.

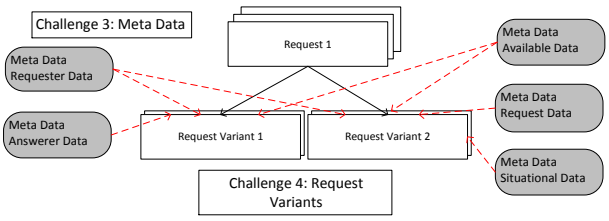


Fig. 3: Data Collection Challenges 3 and 4

3.4 DCC 4: Request Variants

Findings As mentioned, sustainability data exchange in a supply chain involves a considerable number of different manual and automated tasks aligned to the current data request. Hence, execution differs greatly among different data requests, highly influenced by parameters and data and distributed on many sources (cf. DCC 3 and Fig. 3). Moreover, the reuse of provided data is problematic as well as the reuse of knowledge about conducted data requests: persons in charge, managing a data collection, might not be aware of which approach matches the current parameter set.
Problems This makes the whole data collection procedure tedious and error prone. Based on the gained insights, to each data request a data collection process is manually defined initially, and evolves stepwise afterwards. Relying on the various influencing parameters, every request has to be treated individually – there is no applicable uniform approach to a data request, instead a high

number of variants of data collection processes exist. So far, there is no system or approach in place that allows structuring or even governing such varying processes along a supply chain.
Requirements An IS-DCP needs not only to be capable of explicitly defining the process of data collection. Due to the great variability in this domain, it must also be capable of managing numerous variants of each data request relating to a given parameter set. This includes the effective and efficient modeling, management, storage and executing of data collection request processes.

3.5 DCC 5: Incompleteness and Quality

Findings Sustainability data requests are demanding and their complex data collection processes evolve based on delivered data and forwarded requests to other parties (i.e., suppliers of the suppliers) (cf. Fig. 4). Furthermore, they are often tied to regulative requirements and laws as well as involve mandatory deadlines. Therefore, situations might occur, in which not all needed data is present, but the request answer must still be delivered due to a deadline. As another case, needed data might be available, but on different quality levels and/or in different formats.
Problems Contemporary sustainability data collection in supply chains is plagued by quality problems relating to the delivered data. Not only that requests are incompletely answered, the requester also has no awareness of the completeness and quality of the data stemming from multiple answerers. Moreover, answerers have no approach to data delivery in place when being unable to provide the requested data entirely, or their data does not match the request's quality requirements. Missing a unified approach, definitive assertions or statements to the quality of the data of one request can often not be made and requests might even fail due to that fact.
Requirements An IS-DCP must be able to deal with incomplete data and quality problems. It must be possible that a request can be answered despite missing or low quality data. Furthermore, such a system must be able to make assumptions about the quality of the data that answers a request.

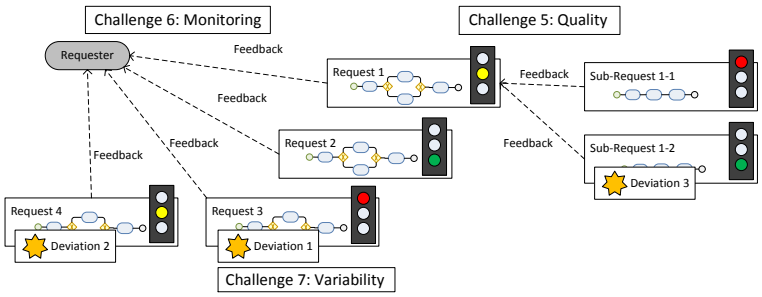


Fig. 4: Data Collection Challenges 5-7

3.6 DCC 6: Monitoring

Findings Sustainability data collection along the supply chain involves many parties and logically may take a long time. The requests exist in many variants and the quality and completeness of the provided data differ greatly (cf. DCC 5). The contemporary approach to such requests does not provide any information about the state of the request to requesters before the latter is answered (cf. Fig. 4). This includes missing statements about delivered data as well as the intermediate requests along the supply chain. If request processing is delayed at the side of one or more answerers, the initial requester cannot access such information without huge effort.
Problems As a requester has no information about the state and potential data delivery problems of his requests, problems only become apparent when deadlines are approaching. However, at that time, it is mostly too late to apply countermeasures to low quality, incomplete data, or answerers that simple deliver no data at all.
Requirements An IS-DCP must be capable of monitoring complex requests spanning multiple answerers as well as various different manual and automatic activities. A requester must have the option to get actively or passively informed about the state of the activities along the data collection process as well as the state of the delivered data.

3.7 DCC 7: Run Time Variability

Findings Data collection requests can take a long time to answer as they dynamically involve a great number of different parties. Further, they expose manual and automatic activities, different kinds of data and data formats, and various unforeseen influences on the data collection process. This implies that parameters, applied at the beginning of the request influencing data collection, may change during the run time of a data collection process. Exceptional situation handling occurs as a result of expiring deadlines or answerers not delivering data.
Problems The variability relating to sustainability data collection processes constitute a great challenge for companies. Running requests might become invalidated due to the aforementioned issues. However, there is no common sense or standard approach to this. Instead, requesters and answerers must manually find solutions to still get requests answered in time. This includes much additional effort and delays. Another issue are external assessments: they could not only be delayed but also completely fail, leaving the answerer without a required certification. The final problem touched by this example concerns mostly long-running data collection processes: data, that was available at the beginning of the query, could get invalid during the long-term process (e.g., if it has a defined validity period).
Requirements An IS-DCP must cope with run-time variability occurring in today's sophisticated sustainability data collection processes. As soon as issues are detected, data collection processes must be timely adapted to the changing situation in order to keep the impact of these issues as considerable as possible.

This requests a system which is able to dynamically adapt already running data collection processes without invalidating or breaking the existing process flow.

4 State of the art

This section gives insights on the state of the art in scientific approaches relating to the issues shown in this paper. It starts with a broader overview and proceeds with more closely related work including three subsections.

Section 3 underlines that exchanging data between different companies along a supply chain in an efficient and effective way has always been a challenge. Nonetheless, this exchange is not only necessary—it is now a crucial success factor and a competitive advantage, these days. However, many influencing factors hamper the realization of a data exchange being automated and homogeneous. In particular for those companies aiming to address holistic sustainability management, the inability to implement automated and consistent data exchange is a big obstacle. Please remind that these companies need to take into account existing and even emerging laws as well as regulations requesting to gather and distribute information about their produced goods. Furthermore, that requested information need be gathered from their their suppliers as well. Hence, complex data collection processes, involving a multitude of different companies and systems, have to be designed, conducted, and monitored to ensure compliance. So far, we could not locate any related work that completely addresses the aforementioned challenges (cf. Section 3).

For complex data collection processes, IS support in the supply chain is desirable supporting communication and enabling automated data collection. The importance and impact of an IS for supply chain communication has already been highlighted in literature various times. In [2], for instance, a literature review is conducted showing a tremendous influence of ISs on achieving effective SCM. The authors also propose a theoretical framework for implementing ISs in the supply chain. Therefore, they identify the following core areas: strategic planning, virtual enterprise, e-commerce, infrastructure, knowledge management, and implementation. However, their findings also include that great flexibility in the IS and the companies is necessary and that IS-enabled SCM often requires major changes in the way companies deal with SCM. As another example, [3] presents an empirical study to evaluate alternative technical approaches to support collaboration in SCM. These alternatives are a centralized web platform, classical electronic data interchange (EDI) approaches, and a decentralized, web service based solution. The author assesses the suitability of the different approaches with regard to the complexity of the processes and the exchanged information. Concluding, the relating work in this area shows or evaluates novel approaches to SCM management, which are, however, mostly theoretic, very general, and not applicable to the specific topic of sustainability data collection processes.

As automation can be a way to deal with various issues for sustainability data collection, various approaches addressing that topic can be found in litera-

ture. However, none of them applies to the domain and specific requirements of sustainable supply chain communication. For example, [4] presents an approach to semi-automatic data collection, analysis, and model generation for performance analysis of computer networks. The approach incorporates a graphical user interface and a data pipeline for transforming network data into organized hash tables and spread sheets for usage in simulation tools. As it primarily deals with a specific type of data transformation, it is not suitable in our context. Such approaches deal with automated data collection; yet they are not related to sustainability or SCM and the problems arising in this setting.

There also exist approaches addressing sustainability reporting (e.g., [5], [6],[7], and [8]). However, they do not suggest technical solutions for automatic data collection. They rather address the topic theoretically by analyzing the importance of corporate sustainability reporting, evaluating sustainability indicators or the process of sustainability reporting as a whole, or aiming at building a sustainability model by analyzing case studies.

Besides approaches targeting generic sustainability, SCM and data collection issues, there are three closer areas that are mainly related to our problem statement and issues. As discussed, sustainability data collection processes involve numerous tasks to be orchestrated. Data requests may exist in many different variants based on a myriad of different data sources and may be subjected to dynamic changes during run-time (cf. DCC 7). This sub-section reviews approaches for process configuration (Section 4.1), data- and user-driven processes (Section 4.2), and dynamic processes (Section 4.3).

4.1 Process Configuration

Behaviour-based configuration approaches enable the process modeler to specify pre-defined adaptations to the process behaviour. One option for realizing this is hiding and blocking as described by [9]. By blocking, this approach allows disabling the occurrence of a single activity/event. The other option enabled by this approach is hiding enabling a single activity to be hidden. That activity is then executed silently but succeeding activities in that path are still accessible.

Another way to enable process model configuration for different situations is to incorporate configurable elements into the process models as described in [10] or [11]. An example of this approach is a configurable activity, which may be integrated, omitted, or optionally integrated surrounded by XOR gateways. Another approach enabling process model configuration is ADOM [12] that builds on software engineering principles and allows for the specification of guidelines and constraints with the process model. A different approach to process configuration is taken by structural configuration, which is based on the observation that process variants are often created by users by simply copying a process model and then applying situational adaptations to it. A sophisticated approach dealing with such cases is Provop [13], which enables process variants by storing a base process models and pre-configured adaptations to it. The later can also be related to context variables to enable the application of changes matching to

different situations. Finally, [14] provides a comprehensive overview of existing approaches targeting process variability.

Process configuration approaches are a promising option to the problem presented in this paper. Nevertheless, that approaches do not completely match the requirements for flexible data collection workflows in such a dynamic and heterogeneous environment, as many different data sources must be considered and request can be subjected to change even while they are running.

4.2 Data- and User-driven Processes

In contrast to classical process management approaches focusing on the sequencing of activities, the case handling paradigm [15] focuses on the objective of the process that is called case. In relation, the product-based workflow approach focuses on the interconnection between product specification and derived workflows [16]. The Business Artifacts approach [17] is a data driven methodology that focuses on business artifacts rather than activities. These artifacts hold the information about the current situation and thus determine how the process shall be executed. In particular, all executed activities are tied to the life-cycle of the business artifacts. Another data-driven process approach is provided by CorePro [18]. It enables process coordination based on objects and their relations. In particular, it provides a means for generating process structures out of the object life cycles of connected objects and their interactions. The creation of concepts, methods, and tools for object- and process-aware applications is the goal of the PHILharmonic Flows framework [19]. Thus, flexible integration of business data and business processes shall be achieved and the limitations known from activity-centered Workflow Management Systems shall be overcome.

The approaches shown in this sub-section facilitate processes that are more user- or data-centric and aware. The creation of processes from certain objects could be interesting for SustainHub, however in the dynamic supply chain environment processes rather rely on context parameters than objects and are also continuously influenced by their changes while executing.

4.3 Dynamic Processes

In current literature, there are two main options for making the automatically supported execution of workflows dynamic: Normal, imperative workflows that are dynamic or adaptive or constraint based declarative workflows that are less rigid by design. This sub-section briefly reviews both kinds of approaches starting with adaptive imperative workflows.

Adaptive PAIS have been developed that incorporate the ability to change a running process instance to conform to a changing situation. Examples of such systems are ADEPT2 [20], Breeze [21], WASA [22], and SPADE [23]. All of these only permit manual adaptation carried out by a user. An important issue in this case is that the exceptional situations leading to the adaptation can occur more than once. In that case, knowledge about the previous changes should be exploited to extend effectiveness and efficiency of the current change [24][25].

In case a human shall apply the adaptations, approaches like ProCycle [26] or CAKE2 [27] aim at supporting him with that knowledge. In the situation described in this paper, these approaches are not suitable since the creation and adaptation of process instances has to incorporate various potentially new information and has to be applied before humans are involved or incorporate knowledge the issuer of a workflow does not possess. Automated creation and adaptation of the data collection workflows will be favourable. In this area, only a small number of contemporary approaches exist, like AgentWork [28] and SmartPM [29] Unfortunately, these are limited to rule based detection of exceptions and application of countermeasures.

As mentioned before, another way to enables flexibility into workflows is by specifying them in a declaring way. By such specification, a strict activity sequencing is not rigidly prescribed. Instead of this, a number of different constraints can be used to specify certain facts that the workflow execution must conform to. This could be the mutual exclusion of two activities or a sequencing relation between two distinct activities. Based on this, all activities specified can be executed at any time as long as no constraint is violated. Examples for such approaches are DECLARE [30] and ALASKA [31]. However, such approaches have specific shortcomings relating to understandability. Furthermore and even more important in our context, if no clear activity sequencing is specified, all activities relating to monitoring are difficult to satisfy and monitoring is a crucial requirement for the industry in this case.

5 Conclusion

This paper motivated the topic of sustainability data exchange along supply chains to subsequently present core challenges as well as state of the art in this area. We have clearly identified seven core challenges for today's data collection processes based on intensive interaction with our SustainHub partners most of them relating to variability issues. Especially, design time as well as run time flexibility are clear requirements for any approach supporting companies aiming at sustainable development and production. The presented challenges can serve as starting point for applications developed to support today's complicated supply chain communication. The challenges are expressed in terms of sustainability data collection, however they describe generic problems that may occur in many domains. Thus the results can be easily transferred and be used for other domains. There exists a substantial amount of related work in different areas touching these topics. Yet, none of these approaches or tools succeeds in providing holistic support for the process of sustainability data exchange in a supply chain. The support of data collection requests and processes along today's complex supply chains is a challenge in the literal sense. Nonetheless, SustainHub is actively working on a process-based solution to deal with, and successfully manage the high variability occurring during design and run time. Future work will describe the exact approach, combination of technologies, and the architecture of the system to cope with the aforementioned challenges.

Acknowledgement

The project SustainHub (Project No.283130) is sponsored by the EU in the 7th Framework Programme of the European Commission (Topic ENV.2011.3.1.9-1, Eco-innovation).

References

1. Fawcett, S.E., Osterhaus, P., Magnan, G.M., Brau, J.C., McCarter, M.W.: Information sharing and supply chain performance: the role of connectivity and willingness. Supply Chain Management: An International Journal **12**(5) (2007) 358–368
2. Gunasekaran, A., Ngai, E.W.T.: Information systems in supply chain integration and management. European Journal of Operational Research **159**(2) (2004) 269–295
3. Pramatari, K.: Collaborative supply chain practices and evolving technological approaches. Supply Chain Management: An International Journal **12**(3) (2007) 210–220
4. Barnett, P.T., Braddock, D.M., Clarke, A.D., DuPré, D.L., Gimarc, R., Lehr, T.F., Palmer, A., Ramachandran, R., Renyolds, J., Spellman, A.C.: Method of semi-automatic data collection, data analysis, and model generation for the performance analysis of enterprise applications (2007)
5. Singh, R.K., Murty, H.R., Gupta, S.K., Dikshit, A.K.: An overview of sustainability assessment methodologies. Ecological indicators **9**(2) (2009) 189–212
6. Ballou, B., Heitger, D.L., Landes, C.E.: The Future of Corporate Sustainability Reporting: A Rapidly Growing Assurance Opportunity. Journal of Accountancy **202**(6) (2006) 65–74
7. Adams, C.A., McNicholas, P.: Making a difference: Sustainability reporting, accountability and organisational change. Accounting, Auditing & Accountability Journal **20**(3) (2007) 382–402
8. Pagell, M., Wu, Z.: Building a more complete theory of sustainable supply chain management using case studies of 10 exemplars. Journal of Supply Chain Management **45**(2) (2009) 37–56
9. Gottschalk, F., van der Aalst, W.M.P., Jansen-Vullers, M.H., La Rosa, M.: Configurable workflow models. Int. J. Cooperative Inf. Syst. **17**(2) (2008) 177–221
10. Rosemann, M., van der Aalst, W.M.P.: A configurable reference modelling language. Information Systems **32**(1) (2005) 1–23
11. La Rosa, M., van der Aalst, W.M.P., Dumas, M., ter Hofstede, A.H.M.: Questionnaire-based variability modeling for system configuration. Software and System Modeling **8**(2) (2009) 251–274
12. Reinhartz-Berger, I., Soffer, P., Sturm, A.: Extending the adaptability of reference models. IEEE Transactions on Systems, Man, and Cybernetics, Part A **40**(5) (2010) 1045–1056
13. Hallerbach, A., Bauer, T., Reichert, M.: Configuration and management of process variants. In: Int'l Handbook on Business Process Management I. Springer (2010) 237–255
14. Torres, V., Zugal, S., Weber, B., Reichert, M., Ayora, C., Pelechano, V.: A qualitative comparison of approaches supporting business process variability. In: 3rd Int'l Workshop on Reuse in Business Process Management (rBPM 2012). BPM'12 Workshops. LNBIP, Springer (September 2012)

15. van der Aalst, W.M.P., Weske, M., Grünbauer, D.: Case handling: A new paradigm for business process support. Data & Knowledge Engineering **53**(2) (2004) 129–162
16. Reijers., H.A., Liman, S., van der Aalst, W.M.P.: Product-based workflow design. Management Information Systems **20**(1) (2003) 229–262
17. Bhattacharya, K., Hull, R., Su, J.: A data-centric design methodology for business processes. In: Handbook of Research on Business Process Management. IGI (2009) 503–531
18. Müller, D., Reichert, M., Herbst, J.: A new paradigm for the enactment and dynamic adaptation of data-driven process structures. In: CAiSE'08. Volume 5074 of LNCS., Springer (2008) 48–63
19. Künzle, V., Reichert, M.: PHILharmonicFlows: towards a framework for object-aware process management. Journal of Software Maintenance and Evolution: Research and Practice **23**(4) (June 2011) 205–244
20. Dadam, P., Reichert, M.: The ADEPT project: A decade of research and development for robust and flexible process support - challenges and achievements. Computer Science - Research and Development **23**(2) (2009) 81–97
21. Sadiq, S., Marjanovic, O., Orlowska, M.: Managing change and time in dynamic workflow processes. Int. J Cooperative Information Systems **9**(1&2) (2000) 93–116
22. Weske, M.: Formal foundation and conceptual design of dynamic adaptations in a workflow management system. In: Proc. Hawaii Int'l Conf on System Sciences (HICSS-34). (2001)
23. Bandinelli, S., Fugetta, A., Ghezzi, C.: Software process model evolution in the SPADE environment. IEEE Transactions on Software Engineering **19**(12) (December 1993) 1128–1144
24. Lenz, R., Reichert, M.: IT support for healthcare processes - premises, challenges, perspectives. Data and Knowledge Engineering **61**(1) (2007) 39–58
25. Minor, M., Tartakovski, A., Bergmann, R.: Representation and structure-based similarity assessment for agile workflows. In: Proc. ICCBR'07. (2007) 224–238
26. Weber, B., Reichert, M., Wild, W., Rinderle-Ma, S.: Providing integrated life cycle support in process-aware information systems. Int'l Journal of Cooperative Information Systems **18**(1) (2009) 115–165
27. Minor, M., Tartakovski, A., Schmalen, D., Bergmann, R.: Agile workflow technology and case-based change reuse for long-term processes. Int'l J. of Intelligent Information Technologies **4**(1) (2008) 80–98
28. Müller, R., Greiner, U., Rahm, E.: AgentWork: A workflow system supporting rule–based workflow adaptation. Data & Knowledge Engineering **51**(2) (2004) 223–256
29. Lerner, B.S., Christov, S., Osterweil, L.J., Bendraou, R., Kannengiesser, U., Wise, A.E.: Exception handling patterns for process modeling. IEEE Trans. Software Eng. **36**(2) (2010) 162–183
30. Pesic, M., Schonenberg, H., van der Aalst, W.M.: Declare: Full support for loosely-structured processes. In: Enterprise Distributed Object Computing Conference, 2007. EDOC 2007. 11th IEEE International, IEEE (2007) 287–287
31. Weber, B., Pinggera, J., Zugal, S., Wild, W.: Alaska simulator toolset for conducting controlled experiments on process flexibility. In: Information Systems Evolution. Springer (2011) 205–221

Enhancing the Case Handling Paradigm to Support Object-aware Processes

Carolina Ming Chiao, Vera Künzle, and Manfred Reichert

Institute of Databases and Information Systems, Ulm University, Germany
{carolina.chiao,vera.kuenzle,manfred.reichert}@uni-ulm.de

Abstract. Despite the widespread adoption of process management systems (PrMS) by industry, there exist numerous processes that cannot be adequately supported by PrMS so far. A common characteristic of these processes, which is usually neglected by traditional activity-centric PrMS, is the role of data as driver for process modeling and enactment. To overcome the limitations caused by missing integration of data and process, several data-centric process management approaches have emerged. A popular one is the Case Handling (CH) paradigm. However, previous case studies pointed out that, although it targets some of the limitations from activity-centric PrMS, the integration of processes and data supported by CH is still unsatisfactory. In this paper, we present the lessons learned from previous case studies and discuss the limitations of CH. We then present the PHILharmonicFlows framework, which enhances the power of data-centric approaches such as CH by enforcing a well-defined modeling methodology governing the object-centric specification and execution of processes and based on a formal operational semantics.

Key words: Case Handling, Data-centric Processes, Object-aware Process Management

1 Introduction

Business process management provides generic methods, tools and techniques for designing, configuring, enacting, and monitoring *business processes* [1]. Existing process management systems (PrMS) are usually *activity-centric*; i.e., processes are defined as a set of "black-box" activities and control flow elements, expressing the order and constraints for executing these activities. However, in these PrMS, business data is typically treated as second-class citizen [2, 3]. Most PrMS only cover atomic data elements, which are needed for control flow routing and as input parameters of process activities. In turn, business objects are usually stored in external databases; i.e., they are outside the control of the activity-centric PrMS. Traditional activity-centric PrMS have been primarily designed for *highly structured*, *repetitive processes*. By contrast, *knowledge-intensive* processes are often *unstructured* or *semi-structured* [4]; i.e., these processes are driven by user decisions and cannot be *straight-jacketed into activities* [2]. Moreover, such

processes require integrated access to data; i.e., users shall be able to immediately access important information at arbitrary points in time during process execution. Additionally, the execution of knowledge-intensive processes depends on the availability of certain information, but not on the completion of a certain activity (as in activity-centric PrMS). Consequently, they are *data-driven*; i.e., instead of depending on activity completion, the progress of process execution depends on changes of correspondent business objects. Besides, these processes normally depend on data from other process instances from the same or different type. Therefore, a PrMS must provide a mechanism for coordinating the interactions between such interdependent processes.

In several case studies in different domains [5, 6, 7, 8, 9], we learned that the described limitations can be traced back to the missing integration of business data and processes. To overcome at least some of the more severe limitations, there exist several approaches that support a tight integration of process and data [2, 3, 7, 10, 11, 12, 13]. One prominent *data-centric* approach is provided by the *Case Handling paradigm* (CH) [2, 14, 15]. In CH, the central concept is the *case* (e.g., an insurance claim or a job offer), which comprises tasks (i.e., activities), data elements, and relations between the tasks making up the process. Although CH overcomes some of the limitations known from activity-centric PrMS, the paradigm is still not broadly used in practice. To better understand the reasons for this, in several case studies [5, 6, 8] we applied the CH paradigm to existing processes. Thereby, we have observed that CH is limited in respect to *object-awareness*. Although CH permits to associate different types of data elements to a case, which may be considered in tight accordance with an *object*, it neither provides explicit support for complex objects nor the relations between them. More precisely, CH supports *object behavior*; i.e., it allows specifying in which order and by whom the data elements (i.e., *object attributes*) shall be written at runtime. However, CH does not properly take into account the *interaction* among different cases or different instances of the same case type. In this paper, we present the lessons learned in these case studies and discuss some of the fundamental limitations of CH. We further discuss the challenges to be tackled to improve the paradigm in order to provide adequate support for object-aware processes. We then give insights into the PHILharmonicFlows framework, which enhance the CH paradigm by giving adequate support to object-aware processes.

Section 2 provides more details on object-aware processes and their characteristics. In Section 3, we present a job application process as example. Along with this example, we present a set of requirements to be met by a PrMS in order to provide an adequate support. In Section 4, CH is introduced followed by a discussion of how CH meets the requirements. Finally, we sketch how to enhance the CH paradigm (and other data-centric approaches as well) by introducing the PHILharmonicFlows framework in Section 5. Section 6 closes with a summary and an outlook.

2 Object-aware Processes

This section describes the fundamental characteristics of *object-aware processes*. Several require a full integration of process and data. As we learned in case studies in a variety of domains [5, 6, 7, 8, 9], object-aware processes present the following major characteristics:

Object behavior. This characteristic deals with the processing of individual object instances. More precisely, for each object type a separate process definition must be provided. At runtime, the latter is then used for coordinating the processing of individual object instances among different users. In addition, it must be specified in which order and by whom the attributes of a particular object instance shall be (mandatorily) written, and what valid attribute settings (i.e., attribute values) are. Furthermore, when executing activities, the involved object instances need to be in certain *states*. Consequently, for each object type, its behavior should be definable in terms of states and transitions. At runtime, the creation of an *object instance* shall be directly coupled with the creation of its corresponding *process instance*. In this context, it is important to ensure that mandatory data is provided during process execution; i.e., during the processing of object instances. For this reason, object behavior should be defined in terms of *data conditions* rather than based on black-box activities.

Object interactions. The behavior of a particular object must be coordinated with the one of other related objects. The related object instances may be created or deleted at arbitrary point in time, resulting in a *complex data structure*. The latter dynamically evolves during runtime, depending on the types and numbers of created object instances. Further, individual object instances of the same type may be in different processing states at a certain point in time. More precisely, it must be possible to execute individual process instances (of which each corresponds to the processing of a particular object instance) in a loosely coupled manner; i.e., concurrently to each other and synchronizing their execution where needed by taking semantic object relations and cardinality constraints into account.

Data-driven execution. To proceed with the processing of a particular object instance, in a given state, certain attribute values are mandatorily required. Hence, object attribute values reflect the progress of the corresponding process instance. More precisely, the setting of certain object attribute values is enforced in order to progress with the process through the use of *mandatory activities*. However, if required data is already available (e.g., it may be optionally provided by authorized users before the respective mandatory activity becomes enabled), these activities will be automatically skipped when being activated. Furthermore, users shall be able to *re-execute* a particular activity, even if all mandatory object attributes have been already set. For this purpose, data-driven execution must be combined with *explicit user commitments*. Finally, the execution of a mandatory activity may depend on attribute values of related object instances. Thus, the coordination of multiple process instances should be supported in a data-driven way as well.

Flexible activity execution. For creating object instances and changing object attribute values, *form-based activities* can be used. Respective user forms should comprise *input fields* (e.g., text fields or check-boxes) for writing selected attributes and *data fields* for reading attributes of object instances. However, different users might prefer different work practices. Activities should therefore be executable at different levels of granularity; e.g., it should be possible that an activity may relate to one or multiple object process instances.

Integrated access. Authorized users should be able to access and manage process-related data objects at any point of time. More precisely, *permissions* for creating and deleting object instances, as well as for reading and writing their attributes need to be defined. Attribute changes contradicting specified object behavior must be prevented. Which attributes may be written or read by a particular (form-based) activity not only depends on the user invoking this activity, but also on the progress of the corresponding process instance. While certain users must execute an activity mandatorily in the context of a particular object instance, others might be authorized to optionally execute this activity; i.e., a distinction is made between mandatory and optional permissions. Furthermore, for object-aware processes, the selection of actors usually not only depends on the activity to be performed, but also on the object instances processed by this activity. In this context, the relationships between users and object instances must be taken into account.

3 Illustrating Example and Requirements

This section presents an example of an *object-aware process* showing the characteristics sketched in Section 2. Following this, we discuss some of the requirements to be met by a PrMS in order to give adequate support to this process.

3.1 Illustrating Scenario: Recruitment Process

As example we consider a (simplified) scenario for recruiting people as known from human resource management (cf. Fig. 1).

Example 1 (Recruitment Process). In the context of recruitment, `applicants` may apply for `job vacancies` via an Internet online form. Before an `applicant` may send her `application` to the respective company, specific information (e.g., `name`, `e-mail address`, `birthday`, `residence`) must be provided. Once the `application` has been submitted, the responsible `personnel officer` in the human resource department is notified. The overall process goal is to decide which `applicant` shall be `invited` for the interview.

When the `personnel` officer receives a `job application`, he may request internal `reviews` for each `applicant`. The concrete number of `reviews` may differ from `application` to `application`. Corresponding `review` forms have to be filled by `employees` from `functional divisions`. `Employees` make a `proposal` on how to

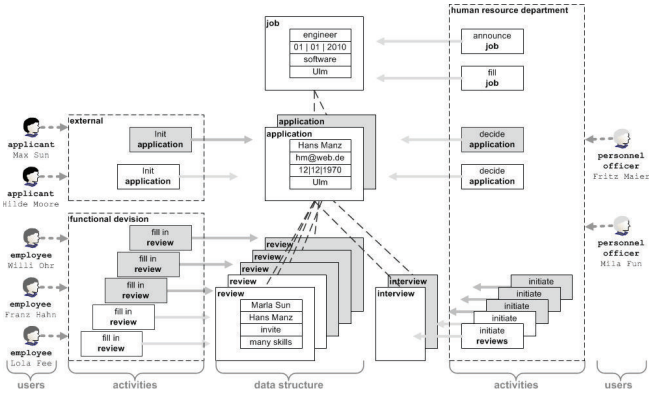


Fig. 1. Example of a recruitment process from the human resource domain

proceed; i.e., they indicate whether the applicant shall be `invited` for an interview or be `rejected`. In the former case an additional `appraisal` is needed. After the `employee` has filled the `review` form, she submits it to the `personnel officer`. Based on the incoming `reviews`, he makes his `decision` on the `application`; i.e., if there are `reviews` indicating the `applicant's interview`, the `personnel officer` shall invite the `applicant` for an interview. Otherwise, the `application` is `rejected`.

3.2 Scenario Requirements

To adequately support such a scenario, any PrMS must meet a set of requirements, which we describe in the following. This will later be followed by a discussion, where we point out which of these requirements are met by CH and which are not.

R1 (Data integration): According to our scenario, the data should be managed in terms of object types comprising object attributes and relations to other object types.

Example R1 (Data integration): For each `job`, a set of `applications` may be created. In turn, for each `application`, several `reviews` may exist. Thereby, a `review` comprises attributes like `application`, `employee`, `remark`, `proposal` and `appraisal`.

R2 (Flexible access to data): Authorized data access should be enabled at any point in time during process execution; i.e., not only during the execution of a particular activity.

Example R2 (Flexible access to data): The `personnel officer` should be allowed to access an `application` even if no activity is currently contained

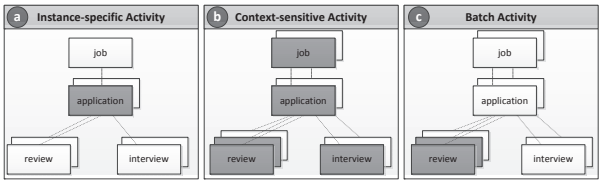


Fig. 2. Types of form-based activities

in his worklist. Furthermore, he should be allowed to update selected attributes from an `application` whenever needed.

R3 (Support of form-based activities and control flow within user forms): A form-based activity comprises a set of *atomic actions*. Each of them corresponds to either an *input field* for writing or a *data field* for reading the value of an object attribute. Which attributes may be written or read in a particular form-based activity may depend on the user invoking this activity and the state of the object instance. In addition, since the writing of particular attributes are mandatory, these forms must signalize which are the corresponding mandatory input fields. However, whether a certain object attribute is mandatory in an activity might depend on the value of other related attributes; i.e., when filling a form, certain attributes might become mandatory on the fly.

Example R3a (Form-based activities): An `employee` requires a form-based activity to write a `review` for an `application`. To complete this activity, she must assign values to attributes `remark`, `proposal`, and `appraisal`. In addition, she might access the attributes values of the corresponding `application`.

Example R3b (Control flow within user forms): If an `employee` chooses to `reject` a `job application`, she must provide a `reason` for this; i.e., attribute `rejection reason` becomes mandatory and a value must be set for it.

R4 (Support of variable activity granularity): Due to the tight integration with data, the behavior of the form-based activities might be related to more than one object instance; i.e., some activities might read/write data in more than one object instance. Accordingly, they may be classified as instance-specific, context-specific, and batch activities. *Instance-specific* activities correspond to exactly one object instance (cf. Fig. 2a). When executing it, attributes of that object instance may be read, written or updated using a form. In turn, a *context-sensitive activity* additionally includes form fields corresponding to higher- or lower-level object instances (cf. Fig. 2b). Finally, *batch activities* allow users to change a collection of object instances in one go; i.e., attribute values are assigned to all selected object instances using one single form (cf. Fig. 2c).

Example R4 (Support of variable activity granularity): An `employee` may choose a *context-sensitive activity* to edit a `review`; i.e., to write attributes `proposal` and `appraisal` and to read attributes referring to the respective `application`. In turn, `personnel officer` may choose a *batch activity* to mark all the other `applications` as "rejected" when an `applicant` is hired for the `job`.

R5 (Support of mandatory as well as optional activities): In order to reach process objectives, certain activities must be mandatorily executed for progressing with the control-flow. At the same time, users should be allowed to optionally execute additional activities; e.g., to write certain attributes even if they are not required at the moment.

Example R5a (Mandatory activity): When performing a `review` of a `job application`, the `employee` must provide a recommendation on whether to `invite` the `job applicant` for an interview or `reject` the `job application`; i.e., a form-based activity needs to be mandatorily executed.

Example R5b (Optional activity): After a `review` has been requested and performed by an `employee`, the `personnel officer` might want to update the `review` request; e.g., attribute `remark` may be optionally updated even if it has been already set.

R6 (Alignment of process execution with object behavior): It should be possible to determine in which order and by whom object attributes must be (mandatorily) written and what valid attribute value settings are. Consequently, for each object type, its behavior should be definable in terms of states and transitions. In particular, it should be possible to drive process execution based on data and to dynamically react on attribute value changes. Hence, it is crucial to map states to attribute values.

Example R6 (Object behavior): The object `review` can be defined by different states: `initiated` (when the `personnel officer` is setting which `job application` is going to be reviewed and which `employee` shall perform the `review`), `under review` (when the `employee` decides whether `reject` or `invite` the `job applicant`), `rejected` (if the `job application` is `rejected`), and `invited` (if the `employee` decides to `invite` the `job applicant` for an interview). An `employee` may only provide a `review` for a particular `job application` if the process is currently at state `under review`. The latter is automatically activated as soon as values for attributes `employee` and `application` have been assigned. If he `rejects` the `job application` (i.e., attribute `proposal` is set as `rejected`), then the attribute `remark` shall instantly become mandatory.

R7 (Support of flexible process execution): The value setting of certain object attributes are mandatory for process execution; i.e., the mandatory activities enforce the value setting of these object attributes as required for progressing with the process. In principle, respective attributes might be written

by executing optional activities as well. If an optional activity is executed before activating a mandatory activity, the latter may be automatically skipped (if other attribute mandatorily set by it have been written before as well).

Example R7 (Flexible process execution): An `employee` might `reject` a `job application`, through an optional activity, while the `personnel officer` is still setting values for other attributes regarding the `review`; i.e., the `review` activity is skipped and will not be shown in the `employee`'s worklist.

R8 (Support of activity re-execution): Users should be allowed to re-execute a particular activity (i.e., to update the referring attributes), even if all mandatory attributes have been already set. To reflect this behavior, users must explicitly commit the completion of the respective activity.

Example R8 (Activity re-execution): An `employee` may change his `proposal` (i.e., `invite` or `reject` the `job applicant`) in the context of a `review` arbitrarily often, as long as he has not explicitly confirmed his decision.

R9 (Enable proper data authorization): To enable access to data at any point in time, permissions for creating and deleting object instances as well as for reading/writing their attributes must be defined. However, attribute changes contradicting to object behavior must be prevented. To achieve this, the progress of the process must be taken into account when granting permissions to change object attributes.

Example R9 (Proper data authorization): An `employee` must not see the `review proposal` of other `employees`. At the same time, she must not update her own `proposal` after submitting it to the `personnel officer`.

4 The Case Handling Paradigm

Case Handling (CH) [2, 14, 15] is a paradigm for supporting flexible and knowledge-intensive processes by strongly integrating them with data. This section first summarizes basic CH concepts. This is followed by a discussion on whether or not CH meets the requirements introduced in Section 3.2.

4.1 Basic Case Handling Concepts

The core concept of the CH approach is the *case type*. The latter comprises tasks (i.e., activities), atomic data elements, and a set of precedence relations between the tasks making up a process. Opposed to traditional activity-centric PrMS, the primary driver for progressing with a case (i.e., a process instance) is not the event related to task completion, but the availability of values for the data elements of the case. While an activity-centric process model clearly separates the process from its associated data, CH integrates both in a tighter manner,

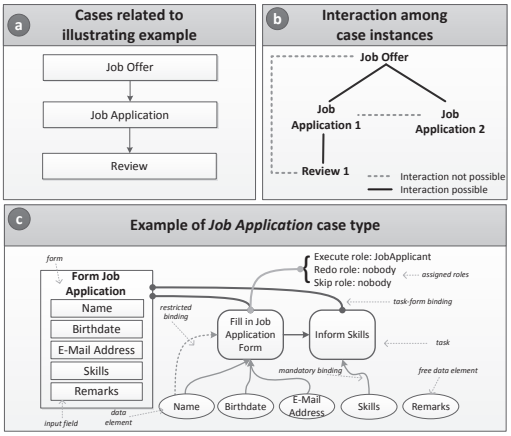


Fig. 3. CH paradigm applied to our example

using produced data not only for routing decisions, but also for determining which parts of the process have already been accomplished. With CH, each *task* may be associated with three sets of *data elements*, each serving a distinct purpose: the first association is between a task and all *data elements* that must be accessible while performing this task. Further, all data elements *mandatory* for a task must be set (i.e., bound to a value) before the task itself is considered to be completed by the CH system. Finally, a data object may have a random number of tasks to which it is *restricted*, meaning that it can only be updated while performing one of these tasks. Interactive tasks are connected to *forms*, each providing access to a selection of data objects. Note that a particular form may be associated with multiple tasks. Finally, it is possible to associate a form to the case itself; i.e., the case and its data elements may be accessed at any point in time using this form.

If a user closes a form after filling out only parts of the mandatory data fields of a task, despite the task not considered as finished, data already entered will still be available to the person who continues working on that task. Such a closely intertwined relation between data and process, however, abandons their often unnatural separation as pursued in traditional PrMS. With the status of data elements being the primary determinant of the case status, this concept overcomes some of the limitations of traditional PrMS:

- Work can now be organized by those performing it with a far higher degree of freedom. Activities may either be performed only partially, without losing intermediary results, or multiple related activities may be handled in one go, surpassing the weakened border between tasks.
- Routing is no longer solely determined by the pre-specified process model. Case types may be designed in such a manner that multiple activities are enabled concurrently, providing different ways of achieving the same goal.

In addition to the *execute role*, specifying the subset of resources allowed to handle a specific task, CH introduces two other roles being crucial for any operational support: the *skip* and *redo* roles. The *skip role* allows users to bypass a selected task. To skip a task, however, all preceding tasks, which have not been completed yet must be skipped (or completed) beforehand. This becomes necessary to ensure an unambiguous state of the process. In turn, the *redo role* enables the user to deliberately roll back the state's case by undoing tasks. In the latter context, the values provided for data objects during the execution of tasks, which are now undone, are not discarded, but merely marked as *unconfirmed*. Hence, they may serve as a default value when re-executing the respective tasks later on. Before a task may be redone, all subsequent tasks that have already been completed must be rolled back as well.

Fig. 3a illustrates an example of how cases and sub-cases can be related. In this example, the case of type `job offer` is related to a varying number of sub-cases of type `job application`. In turn, this sub-case may be related to different `reviews`. Details of a case type are shown in Fig. 3c. Thereby, data elements are associated with forms representing corresponding input fields. In turn, forms are associated with tasks. In our example, the depicted form is associated to both tasks of the case. More precisely, data elements `name`, `birthdate` and `e-mail address` are mandatory for completing task `Fill in job application form`, while data element `skills` is mandatory for task `Inform skills`. In particular, data element `name` may be only filled in the context of task `Fill in job application form`. Data element `remarks`, in turn, is a so-called *free data element*; i.e., it may be accessed at any point in time during case executiong by any involved user role.

4.2 How Does Case Handling Meet the Requirements of Object-aware Processes

We now analyze and discuss whether CH meets the requirements of object-aware processes presented in Section 3.2. This analysis is based on previous case studies [5, 6, 8] as well as on hands-on experience with the CH tool BPMone[1]. The latter is a commercial tool that implements the concepts of the CH paradigm.

R1 (Data integration): In CH, a *case* may be considered in tight accordance with an *object*; i.e., the data elements related to a case may logically be considered as data attributes of an object. Further, a case may be hierarchically related with *sub-cases*, representing object relationships. However, only direct relations (i.e., case and correspondent sub-cases) are supported. Neither interactions between two instances of the same case nor access to a case's data elements by a corresponding sub-case are allowed (cf. Fig. 3b).

R2 (Flexible access to data): One of the main characteristics of the CH paradigm is its focus on the *entire case*; i.e., all users get full reading access to the whole case when they are executing any activity. In particular, *context*

[1] http://www.perceptivesoftware.com/products/perceptive-process/business-process-management

tunneling (i.e., limitation of the user's view to single work items) is avoided, which increases process flexibility. The users involved in one case instance may read all related data elements of the case at any point in time. Additionally, values of *free data elements* may be edited by all users at any point in time during case execution.

R3 (Support of form-based activities and control flow within user forms): Like activity-based modeling approaches, in CH, process steps correspond to activities. Such activities can be implemented in terms of *forms*. Each form is linked with a collection of atomic data elements, which are either mandatory or restricted. An activity will be considered as completed if all mandatory data elements have an assigned value. However, CH does not allow a field (and the corresponding data element) to become mandatory dynamically; i.e., there are no mechanisms that allow defining the internal flow logic within a form. Hence, this requirement is not completely met.

R4 (Support of variable activity granularity): Similar to activity-centric approaches, in the CH paradigm, each task instance is related to exactly one process instance (i.e., instance-specific activities). However, using sub-cases as workaround, context-sensitive activities are partially; i.e., if a case has sub-cases, its tasks may access data elements from its sub-cases. Since CH does not support interactions among different instances of the same case type, batch activities are not directly supported.

R5 (Support of mandatory as well as optional activities): In CH, a task may have associated data elements that must be set before completing the case. Setting these *mandatory* data elements is considered as a mandatory activity. In turn, the definition of optional activities in CH is enabled by the use of free data elements. Since the latter are not relevant for process control, they may be set at any point in time during case execution by any user involved in the case.

R6 (Alignment of process execution with object behavior): In particular, the data elements of a case may be associated with tasks (i.e., activities) and be declared as *mandatory* or *restricted*. *Free* data elements, in turn, may be changed at any point in time by any user involved in the case. Thereby, CH allows capturing of the *object behavior*. In other words, by associating mandatory data elements with tasks from a case, it becomes possible to specify in which order and by whom these data elements (or object attributes) shall be written. Additionally, input fields and transitions may be associated with constraints on attribute values. Finally, it is possible to initialize and terminate a case instance at any point in time.

R7 (Support of flexible process execution): Enabling of a task is driven by data; i.e., tasks are enabled when data becomes available. When all mandatory data elements, related to a particular activity, are set, the subsequent activity becomes enabled. Additionally, tasks may be automatically skipped at runtime if their mandatory data elements have been provided by other tasks before.

R8 (Support of activity re-execution): For each task in CH, separate roles can be defined. More precisely, it is possible to define who shall work on

an activity and who may redo or skip it. The redo role allows actors to execute activities several times. However, the user may re-execute the task only until the completion of the respective case. If the execution of the case is completed, the user must not re-execute the task anymore. Hence, *user commitments* are not considered; i.e., the user cannot indicate to the system that he agrees with the values, set for the data elements, so that the task is then finished in a user-controlled way.

R9 (Enable proper data authorization): In CH, data elements are associated with the tasks in order to define the order in which they shall be set. The user role assigned as *execute* role to a particular activity is the one allowed to write the related data elements. However, the direct association of roles with tasks does not allow declaring such tasks as *optional* for other user roles. Moreover, to avoid *context-tunneling*, any user involved in the case may read information of the *entire case* at any point in time; i.e., data privacy becomes a severe issue.

5 Enhancing the Case Handling Paradigm

The CH paradigm overcomes many of the limitations caused by the missing integration of data and processes. However, as discussed, there still exist requirements not fully met by CH; e.g., concerning flexible activity granularity and data privacy issues. Although CH focus on data as driver for process execution, it does not take object *states* and *transitions* between them into account; i.e., CH does not enable mappings between attribute values and objects states and therefore is unable to ensure compliance between them. To deal with these drawbacks of the CH paradigm and other existing data-centric approaches, we have developed the PHILharmonicFlows framework. Fig. 4 summarizes its main components. Basically, PHILharmonicFlows comprises both *modeling* and *runtime environment* enabling full lifecycle support for object-aware processes. As a fundamental prerequisite, a *data model* describing the respective domain needs to be defined; i.e., object type and relations are defined (cf. Fig. 5a).

The *modeling environment* of PHILharmonicFlows enforces a well-defined modeling methodology governing the definition of processes at two different levels of granularity: *micro* and *macro processes*. A micro process captures the *behavior* of an object (cf. Fig. 6), while the macro process realizes the *objects interactions*. In particular, PHILharmonicFlows supports the interaction and coordination of object instances asynchronously as long this does not violate any semantic dependencies to be considered.

Process and data authorization is based on *user roles*. Data may be accessed optionally and at any point in time. In turn, process execution is based on permissions for creating and deleting object instances as well as for reading or writing their attributes. Furthermore, to enable access at the object attribute level, PHILharmonicFlows maintains a comprehensive *authorization table*. Finally, based on these authorization settings, PHILharmonicFlows automatically generates user forms. Besides the form-based activities, the framework support

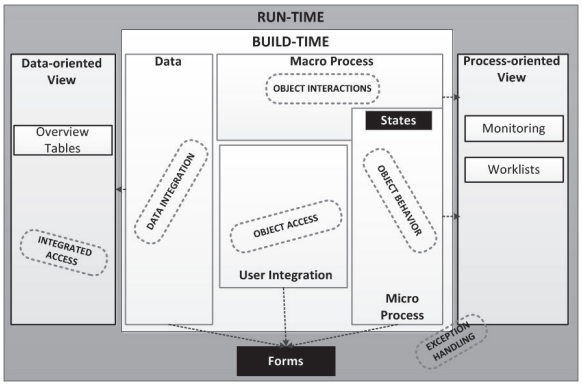


Fig. 4. PHILharmonicFlows

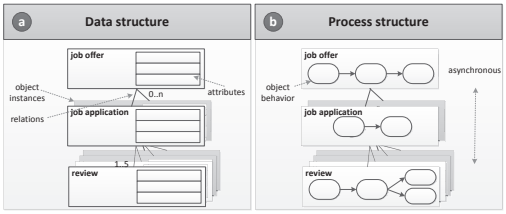


Fig. 5. Data structure and corresponding process structure

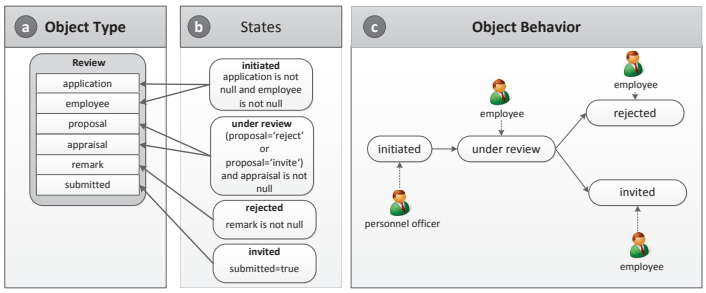


Fig. 6. Object behavior defined based on states and transitions by PHILharmonicFlows

black-box activities as well (i.e., activities that invoke external applications or functions that implement more complex computations).

The *runtime environment* provides *data-* and *process-oriented* views to end-users; i.e., authorized users may invoke activities for accessing data at any point in time as well as activities needed to proceed with the flow of the processes. Moreover, PHILharmonicFlows is based on a well-defined formal semantics, which allows for the automatic generation of end-user components corresponding to the runtime environment (e.g., user worklists and form-based activities).

Due to the lack of space, we do not describe the components of the framework. A more detailed description of the framework as well as its components can be found in [7, 16].

6 Summary & Outlook

In this paper, we discussed the limitations of the CH paradigm regarding the support of object-aware processes. We based this discussion on several case studies, including hands-on experience we gathered when working with the CH system BPMone.

+ *supported* o *partially supported* - *not supported*	Case Handling Paradigm	PHILharmonicFlows Framework
R1 (Data integration)	o	+
R2 (Flexible access to data)	+	+
R3 (Support of form-based activities and control flow within user forms)	o	+
R4 (Support of variable activity granularity)	o	+
R5 (Support of mandatory as well as optional activities)	+	+
R6 (Alignment of process execution with object behavior)	+	+
R7 (Support of flexible process execution)	+	+
R8 (Support of activity re-execution)	o	+
R9 (Enable proper data authorization)	-	+

Fig. 7. Comparing the CH paradigm and PHILharmonicFlows

In particular, CH does not enable mappings between attribute values and objects states and therefore is unable to ensure compliance between them. Hence, problems like the limited interaction among case instances and lack of data privacy make the CH paradigm unsuitable for object-aware processes. We further presented the PHILharmonicFlows framework. Its objective is to enhance the power of CH paradigm and other data-centric approaches to enable proper support of such processes. This approach has been applied in several case studies comprising different domains (e.g., human resources management, healthcare, and automotive industry). As proof-of-concept, a prototype has been developed, enabling the modeling and enactment of object-aware processes. Finally, Fig. 7 shows a compilation of the requirements of object-aware processes and how well both approaches (i.e., CH and PHILharmonicFlows) meet them.

Acknowledgements

The authors would like to acknowledge the financial support provided by the Ernst Wilken Foundation.

References

[1] Reichert, M., Weber, B.: Enabling Flexibility in Process-aware Information Systems: Challenges, Methods, Technologies. Springer (2012)

[2] van der Aalst, W.M.P., Weske, M., Grünbauer, D.: Case handling: A new paradigm for business process support. Data & Knowledge Engineering **53(2)** (2005) 129–162

[3] Cohn, D., Hull, R.: Business artifacts: A data-centric approach to modeling business operation and processes. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering **32(3)** (2009) 3–9

[4] Silver, B.: Case management: Addressing unique bpm requirements. Technical report, BPMS Watch (2009)

[5] Künzle, V., Reichert, M.: Towards object-aware process management systems: Issues, challenges, benefits. In: Proc. BPMDS'09. (2009) 197–210

[6] Künzle, V., Reichert, M.: Integrating users in object-aware process management systems: Issues and challenges. In: Proc. BPM'09 Workshops. (2009) 29–41

[7] Künzle, V., Reichert, M.: Philharmonicflows: Towards a framework for object-aware process management. Journal of Software Mainteinance and Evolution: Research and Practice **23**(4) (2011) 205–244

[8] Künzle, V., Weber, B., Reichert, M.: Object-aware business processes: Fundamental requirements and their support in existing approaches. Int'l Journal of Information System Modeling and Design **2**(2) (2011) 19–46

[9] Chiao, C.M., Künzle, V., Reichert, M.: Towards object-aware process support in healthcare information systems. In: Proc. eTELEMED 2012. (2012) 227–236

[10] Bhattacharya, K., Hull, R., Su, J.: A data-centric design methodology for business processes. In: Handbook of Research on Business Process Management. (2009) 503–531

[11] Liu, R., Bhattacharya, K., Wu, F.Y.: Modeling business contexture and behavior using business artifacts. In: Proc. CAiSE'07. (2007) 324–339

[12] Müller, D., Reichert, M., Herbst, J.: Data-driven modeling and coordination of large process structure. In: Proc. CoopIS'07. (2007) 131–149

[13] Vanderfeesten, I., Reijers, H.A., van der Aalst, W.M.P.: Product-based workflow support: Dynamic workflow execution. In: Proc. CAiSE'08. (2008) 571–574

[14] Guenther, C., Reichert, M., van der Aalst, W.M.P.: Supporting flexible processes with adaptive workflow and case handling. In: Proc. ProGility'08. (2008) 229–234

[15] Mutschler, B., Weber, B., Reichert, M.: Workflow management versus case handling: results from a controlled software experiment. In: Proc. SAC'08. (2008) 82–89

[16] Künzle, V.: Object-aware Process Management. PhD thesis, Ulm University, Germany (2013)

Knowledge and Business Intelligence Technologies in Cross-Enterprise Environments for Italian Advanced Mechanical Industry - *Project Presentation* -

Francesco Arigliano[3], Antonia Azzini[1], Chiara Braghin[1], Antonio Caforio[2], Paolo Ceravolo[1], Ernesto Damiani[1], Vincenzo Savarino[3], Claudia Vicari[3], and Francesco Zavatarelli[1]

[1] SESAR Lab - Dipartimento di Informatica
Università degli Studi di Milano, Italy
Email: {name.surname}@unimi.it
[2] Centro Cultura Innovativa d'Impresa, Universit del Salento, Lecce, Italy
Email: {name.surname}@unisalento.it
[3] Research & Development Laboratory - Engineering, Ingegneria Informatica, Italy
Email: {name.surname}@eng.it

Abstract. The internetworking and the outsourcing of business activities have become essential in the short term to maintain competitiveness in the market, as it allows to significantly reduce time and cost of core business processes. However, outsourcing, increasing the level of information sharing, imposes new precautions to maintain, in the medium and long-term, strategic control over the knowledge produced and exchanged, both in terms of "know-how" and "know-that". Especially for SMEs, his implies substantial risk of a technological nature, in that it requires complex and extremely expensive technological framework. KITE.it aims to develop a methodological and technological framework to support the transition of the advanced mechanical supply chains towards Value Network models able to guarantee:
– interoperability and cooperation between firms and individuals in the network;
– the management and securing of the intellectual capital;
– measurement and performance optimization.

Key words: Business Process Management, Data Analysis

1 Introduction

The exit from the great global crisis towards a new cycle of development requires to move from organizational and inter-organizational models, based on a strict definition of roles and organizational boundaries, to structures defined as a Value Network (VN): an organizational structure by fluid boundaries and the complex relational dynamics in which individuals, groups and organizations thrive through complex processes of interchange and integration of value, based

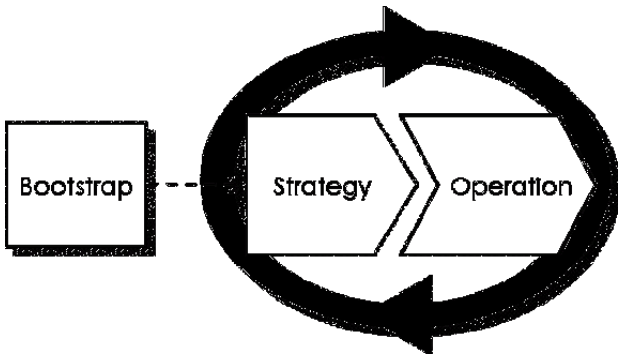


Fig. 1. KITE.it end to end methodology

on the network paradigm[2]. In this context, the competitiveness of the Made in Italy must be defended and enhanced redefining business models and business processes according to the VM Paradigm. We define VN as the integration of a Business Network (or network of enterprises) and the corresponding Social Network: the first characterized by the mediation of the economic value, while the second by the mediation of knowledge and intellectual capital of knowledge workers. In an increasingly, uncertain and changeable market, business agility is the mantra of knowledge driven organization [1]. A variety of tools and technologies were developed to simplify the communication among organizations and people across the VN. These tools are focused on information sharing and are characterized by the ability to integrate information systems, to connect the processes of an organization to those of suppliers and make the process accessible to the customers. Exhortation to collaboration, sharing, cooperation and the ability to rapidly set up their business and the value network in which an organization operates, is hampered by several kinds of issues, such as the dissemination of know-how, and this could damage the company. KITE.it project, is aimed at providing the conceptual, methodological and technological tool to maximize the ability to obtain agile, collaborative and social business in a secure manner, that is minimizing the risk. Therefore the fundamental ambition is the safe business agility; which means that both the process and the entire organization needs to be flexible.

2 KITE.it Methodology

An agile organization is expected to adapt itself to a changing environment proactively. Such adjustment should be done quickly at the level of modeling and implementation: a modified model is to be transferred seamlessly and quickly to the computer systems supporting the organization.

KITE.it "End to End" Methodology manages iteratively the entire business life cycle both at strategic and operational level. At the strategic level the exogenous variables and the value network in which the organization operate are

Fig. 2. Elements of the Strategy phase

analyzed. At the operational level, the strategy and the corporate policies are realized by an architecture of core processes supported by support processes. The big picture of KITE.it methodology is very simple. As described in Fig.1, it consists of two main iterative phases, which correspond to the two levels of analysis the methodology is based on: Strategy and Operations, preceded by a initial bootstrap phase in which teams are set up and the methodology implementation plan is established. As described in Fig.2, in the Strategy phase we identify four iterative steps:

- Strategic Analysis S.
- Goal Setting S2.
- KPI and Target S3.
- Risk and Policies S4.

2.1 Strategical Analysis S1

After an analysis of the value network environment (socio-economic-political), it is necessary:

- to establish the vision,
- to analyze the value networks in which the company operates by identifying roles and value flows,
- and finally to define the value proposition.

The E3Value method [3] is indicated by KITE.it methodology as the preferred notation to model Value Networks. In addition, services to be implemented or modified are identified. The last step of this phase is to establish the value chain, that is the core processes that will be linked with the goals identified in step S2.

2.2 Goal Setting S2

At this stage strategic goals, that the organization wants to achieve, are identified. This analysis is done according to four different perspectives:

- Financial.
- Value Network.
- Processes.
- Learning and growth.

These perspectives are borrowed from the BSC methodology [4] (balanced scorecards) in which the Customer perspective is extended to the Value Network and the Internal Process perspective is extended to cover collaborative processes.

2.3 KPI and Target S3

All goals previously identified are mapped to key performance indicators and target values, that give the possibility to check the distance from each goal achievement. Trough the objective identified in the scorecard perspectives, at this stage we establish all the measures with a low level of detail. In the operational phase methodology such measures will become detailed process metrics.

2.4 Risk and Policies S4

The final step in this phase gathers business requirements for the operational phase, policies that will support the objectives are established and the business risks associated with the objectives and any policies to mitigate them are identified. The Strategy phase is iterative by nature and therefore the steps described are repeated until a stable and shared strategic model is obtained. These iterations may affect structural changes: once a risk is identified and its probability and impact is assessed, it may be necessary not only to review the objectives and the policies, but sometimes even the value proposition with profound effects on the organization. In order to carry out this phase in a truly effective way, it is needed the active involvement of the top management. As described in Fig.2, in the Operation phase we identify four iterative steps:

- Business modeling - O1.
- Define Metrics - O2.
- Enacts - O3.
- Monitoring - O4.

2.5 Business Modeling - O1

This is the stage where the requirements of the strategic analysis become business models or diagrams; different models will be defined to establish the process architecture, the organizational structure, processes at various levels of detail, the business decisions and the operational risks associated with the processes.

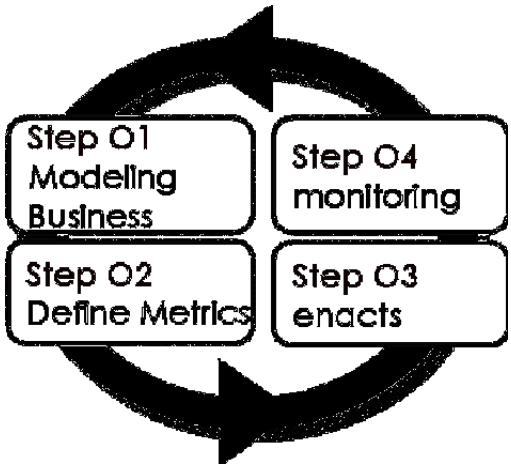


Fig. 3. Elements of the Operation phase

2.6 Define Metrics - O2

Starting from performance indicators, identified in the step S3 of the Strategy phase, the measures to be carried out on individual processes and methods for the recovery of the necessary information are set out in detail. The KITE.it methodology provides a model for the metrics specifically defined by the project.

2.7 Enacts - O3

At this stage the models are ingrained into the organization's operations. The necessary components will be developed and put into production with the processes by integrating all in an environment of social cooperation.

2.8 Monitoring - O4

The Monitoring stage is critical to ensure the ability:

– to continuously improve the performance of the organization,
– to verify the achievement of strategic objectives,
– and to control risks.

Using the information about the process, and the security and SNA measures, we will be able to close the loop and to reiterate the end-to-end methodology by restarting from the Strategic analysis (S1) or from Business modeling (O1) to make ever more effective the action of the business.

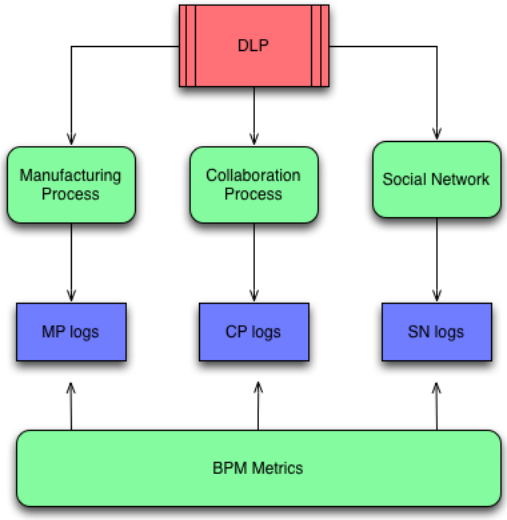


Fig. 4. Kite.it integrated BPM approach

3 Data Loss Prevention Scenario

Our first investigations was related to a Data Loss Prevention Scenario. The loss of sensitive information are critical for organization, solutions for preventing data leakage incidents are based on systems designed to detect potential data breach transmissions and prevent them by blocking data while in-use (endpoint actions), in-motion (network traffic), and at-rest (data storage)[5]. These systems provides logs describing the interactions among the organization and are typically able to track the originator and addressee of a data transmission, together with the action operated on data. Using this information KITE.it tacks the dynamics on the exchange of intellectual capital within the VN. In fact, the DLP system allows to extract information about the manufacturing process, the collaboration process and the social network of the interactions. The objective of KITE.it is to provide an unified view on these dimensions providing integrated metrics to enhance Business Process Monitoring, as illustrated in Fig.4.

Acknowledgements

This work was partly funded by the Italian Ministry of Economic Development under the Industria 2015 contract -KITE.it project.

References

[1] Allee, V.: The future of knowledge: Increasing prosperity through value networks. Routledge (2003)

[2] Prahalad, C.K., Ramaswamy, V.: The future of competition. Harvard Business School Press, Boston, MA (2004)

[3] Gordijn, J., Akkermans, H., Van Vliet, J.: Designing and evaluating e-business models. IEEE intelligent Systems **16**(4) (2001) 11–17

[4] Kaplan, R., Kaplan, R.S., Norton, D.P.: The balanced scorecard: translating strategy into action. Harvard Business Press (1996)

[5] Shabtai, A., Elovici, Y., Rokach, L.: A taxonomy of data leakage prevention solutions. In: A Survey of Data Leakage Detection and Prevention Solutions. Springer (2012) 11–15

On Process Rewriting for Business Process Security
(Extended Abstract)

Rafael Accorsi

University of Freiburg, Germany
accorsi@iig.uni-freiburg.de

Abstract. This paper reports on ongoing work towards a framework to automatically rewrite business process models and, thereby, correctively enforce adherence to regulatory compliance and security policies. Specifically, the paper first motivates the need for rewriting mechanisms as a means to enforce a particular class of properties. Second, it describes the main building blocks of ReWrite, the framework to automatically rewriting process specifications. Third, in order to preserve the functional goals of the processes upon rewriting, a set of congruence relations is defined and their appropriateness is discussed. The presentation of the formal framework and rewriting algorithms is deferred to the full paper version.

1 Introduction

Ensuring the compliance of business processes to regulations and security policies is of utmost importance in business process management [1, 31]. Approaches to assess compliance can be classified into [8, 25]: (1) *forward compliance checking* aims to design and implement processes where conforming behavior is enforced and (2) *backward compliance checking* aims to detect and localize non-conforming behavior. This paper focuses on forward compliance checking based on business process models and policies. In this setting, previous work addresses the annotation of business processes [15, 23], requirements elicitation [11, 10, 26] and formal verification [2–4, 6]. None of the previous work has considered the intersection of rewriting techniques for programs and the corrective enforcement of business processes compliance. This paper sets out to investigate this connection.

Specifically, this paper starts by motivating the need for rewriting mechanisms as a means to enforce a class of complex safety properties which encompass compliance requirements. In contrast to previous works, which merely present the challenge of enforcing compliance, this paper formally substantiates the need for rewriting using formal enforcement theories of computer security. By doing so, one establishes the relationship between the class of properties and the corresponding enforcement mechanisms needed to guarantee these properties.

Based on this, the paper further presents the building blocks and examples of the framework ReWrite to automatically rewrite business processes specifications and, thereby, ensure compliance with pertinent policies. The overall approach is

depicted in Fig. 1. The key insight is to split responsibilities during the modeling phase and focus on the core competencies of each actor: the business experts design a process model (e.g. using BPMN or BPMN) and compliance experts design the compliance policy (i.e. description of applicable controls) and denote the processes to which they apply. The goal of ReWrite is to take process and policy specifications as input and produce an executable process model which complies to the policy.

Clearly, modifying the structure of the process may lead to a specification that does not achieve the business goals designated for that process. To circumvent this issue while still allowing for rewriting, one must define similarity relations between the original and the modified processes in terms of execution traces their produce: only modifications that preserve the similarity relation are appropriate. These relations are, however, not characterized in the literature. This paper defines and investigates the adequacy of four congruence relations. Initial investigations carried out with ReWrite show that rewriting operators – append, delete and replace activities in process models – can be supported by such relations, thereby allowing for effective rewriting algorithms.

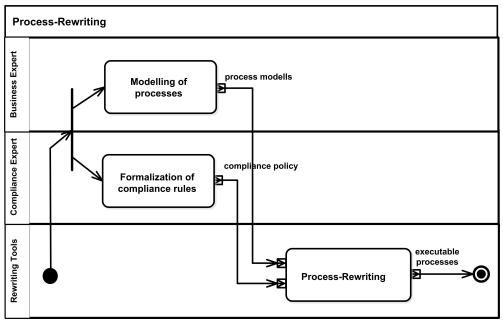


Fig. 1. ReWrite approach.

Taking stock, this paper provides the following contributions:

- Establishes the relationship between the compliance requirements, the underlying formal properties and the enforcement mechanisms. In particular, it shows a class of properties that can only be enforced using process rewriting and that process rewriting can enforce all other kinds of properties.
- Presents the main building blocks of ReWrite. Specifically, it presents μBPMN, a Turing complete fragment of BPMN to express business processes and a meta-model for the expression of compliance policies. An example will illustrate the interplay of these requirements for rewriting.
- Defines four congruence relations to guarantee the compliance with requirements while preserving the business (functional) goal of the process and discusses their adequacy for rewriting.
- Reports on ongoing implementation of the ReWrite framework "as-a-service" and describes how its evaluation will be conducted.

The rewriting framework proposed in this paper can be used in three dimensions and timepoints along the business process management lifecycle. Firstly, *before* the process execution to ensure compliance "by design". Here, if one assumes that process are stable and faithfully executed by the business process management system, then rewriting is capable of enforcing the designated class

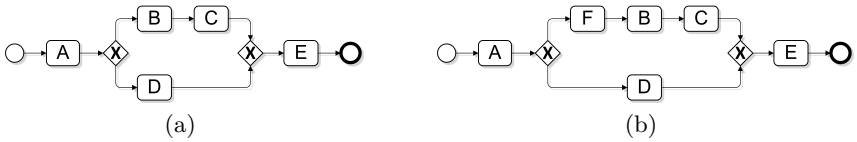


Fig. 2. Original process (a) and process after rewriting (b) .

of compliance properties. Secondly, *during* the execution for corrective enforcement. In this setting, an execution monitor with rewriting capabilities is needed to enforce properties. While this dimension comes along with a considerable performance overhead, it allows for flexible process whose compliance can be guaranteed at runtime. Thirdly, *after* the execution as a mechanism for automated process enhancement and re-design based upon process models reconstructed from event logs using process mining [30].

While the foundation of ReWrite can be used in any of these perspectives and timepoints, this paper focuses on the "by design" perspective – especially in terms of implementation. We currently apply these techniques to re-design reconstructed processes. Carrying over these concepts to in-line process monitoring is subject of future work. Further, it should be remarked that the focus of this paper is the presentation of the ReWrite framework, its building blocks and application. Pertinent proofs for properties – e.g., the Turing completeness μBPMN – are deferred to a longer version of the paper.

Paper structure. Section 2 establishes the relationship between classes of compliance properties and the corresponding enforcement mechanisms, thereby motivating the need for rewriting. Section 3 introduces the main building blocks of the rewriting approach, i.e. the process language and policies. Section 4 introduces the congruence relations and the corresponding rewriting operations. Section 5 reports on the implementation and evaluation of the approach.

Example 1. We illustrate the high-level operation of ReWrite using an example. Consider the process model in Fig. 2(a). It stands for a medical workflow, namely updating the patient record. Activity A denotes a staff member inputting the `patient_ID`. If the `ID` exists, B queries the database and C updates the database with the updated patient record; otherwise, if there is no such an `ID`, D issues an error message. In both cases, E deletes local copies of query and record. One security policy may require that, *before* showing the query results, the staff member must be authenticated and authorized to do so, which is encoded with the activity F. Given that information, rewriting would inspect the original model to detect whether F is at the correct place. If not, ReWrite will add F to the process, which produces the process model depicted in Fig. 2(b). The remainder of this paper shows how to carry out such a rewriting. ⊣

2 Properties and Enforcement Mechanisms

Non-functional properties – be it security or privacy properties, or properties arising from compliance requirements – can be classified according to two hierarchies: (1) the *Safety-Progress* based on languages used to *formalize* the properties [12] and (2) the *Safety-Liveness* based on mechanisms used to *enforce* the properties [7]. This section argues that the enforcement of compliance requirements demands process rewriting. It does so by revisiting these two theories.

2.1 Safety-Progress Hierarchy

Chang et al. [12] define four operators and, based upon them, six formal languages organized in a hierarchy. Let Σ^* be the set of all finite words over an alphabet Σ, Σ^+ the set of non-empty, finite words and Σ^w the set of all infinite words over the alphabet Σ. Let Φ be a finite language, the operators are:

- $A(\Phi)$ encompasses all σ, s. t. *every* prefixes of σ are in Φ.
- $E(\Phi)$ encompasses all σ, s. t. *some* prefixes of σ are in Φ.
- $R(\Phi)$ encompasses all σ, s. t. *infinite many* prefixes of σ are in Φ.
- $P(\Phi)$ encompasses all σ, s. t. *all but a finite number* of prefixes of σ are in Φ.

With these operators at hand one can define the six languages which build, for finite languages Φ and Ψ, the hierarchy depicted in Fig. 3.

1. *Safety language*: $\Pi = A(\Phi)$.
2. *Guarantee language*: $\Pi = E(\Phi)$.
3. *Obligation language*: $\Pi = \cap_{i=1}^{m}(A(\Phi_i) \cup E(\Psi_i))$.
4. *Response language*: $\Pi = R(\Phi)$.
5. *Persistence language*: $\Pi = P(\Phi)$.
6. *Reactivity language*: $\Pi = \cap_{i=1}^{m}(R(\Phi_i) \cup P(\Psi_i))$.

In Fig. 3, each language encompasses those beneath it. This follows directly from the definition of the languages. For example, the language *Obligation* is built by the conjunction of the languages *Safety* and *Guarantee*. Further, all the *Safety* languages can be expressed in terms of the *Obligation* language, wheras the corresponding *Guarantee* part is empty (i.e. $E(\Psi_i) = \emptyset$)

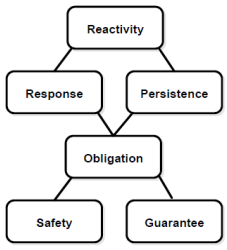


Fig. 3. Safety-response.

Obligation and Guarantee. Due to the existence operator $E(\Phi)$ the *Obligation* and *Guarantee* languages are relevant for the formalization of security and compliance requirements. For example, classical access control could be expressed in a way that the required prefix – denoting the process state – is available already by during the access decision. In this case, it can be decided that the property holds and could be no longer violated.

In another case, the required prefix is not at the decision time and must be built during the further continuation of the process instance. Here, one cannot decide whether the obligation will be fulfilled in a later point in time. The enforcement of such property with classical access control is not possible. That is because it cannot be decided whether the obligation will be fulfilled or not. If at a given timepoint t one is to evaluate where a process instance is in the *Obligation* language O_1, then four different states are possible [20]: (1) *true* it can be shown that the instance is in O_1; (2) *possibly true* at t the instance is in O_1, but there is the possibility that the whole instance will not be in O_1; (3) *possibly false* at t the instance is *not* in O_1, but it is possible that will eventually be in the language; finally, (4) *false*, it can be shown that the instance is not in O_1.

The rewriting approach presented in this paper allows for the automated enforcement of cases that evaluate to *possibly false*. The other cases can be tackled with existing access control mechanisms, in that they on the one hand prevent the transition from *possibly true* to *false* or *possibly false*, or if they do not allow the execution of a *false* instance.

2.2 Safety-Progress Hierarchy

This hierarchy is defined upon three well-known classes of preventive enforcement mechanisms, namely: (1) *static verification* before the execution; (2) *runtime monitoring* during the execution; and (3) *rewriting* before or during the execution. In particular, each of these mechanisms can recognize and therefore circumscribe a particular class of properties. The following formally defines the hierarchy, which is depicted in Fig 4.

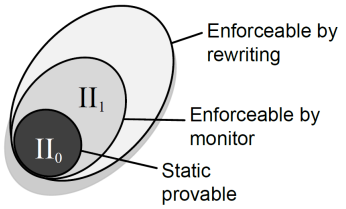


Fig. 4. Safety-liveness hierarchy.

Class Π_0: Statically enforceable properties. A policy $\mathcal{P}$ is statically enforceable if there is a Turing machine $M_{\mathcal{P}}$ for $\mathcal{P}$ that terminates in finite time if $\mathcal{P}$ holds in the process. It can be shown that the class of enforceable security properties corresponds to the class of recursively decidable properties of programs, which in turn corresponds to the arithmetic class Π_0. (See [16] for details.) Several properties can be statically verified before the execution of a process [17]. The proof of a property stating that, e.g., a (stable) process calls exactly 30 services is trivial: enumerate and count the service calls.

Class Π_1: Runtime properties. The basis for the formalization of properties of programs which can be enforced during the execution (so-called "execution monitoring") is an execution machine that generates traces (sequences of events). Given a trace, for properties that can be enforced with an execution monitor there must be a detector that identifies their violation during the runtime. Such a detector must exhibit the following properties [28]: (1) the property detected by the detector is irremediably violated; (2) the detector must identify the violation in finite time; and (3) the detector mus be recursively decidable.

The following criteria thus formally define the detector:

$$\forall \Pi \in \Psi(\Pi) : \mathcal{P}(\Pi) \Rightarrow \forall \sigma \in \Pi : \hat{\mathcal{P}}(\sigma) \tag{1}$$

$$\forall \tau' \in \Psi : \neg\hat{\mathcal{P}}(\tau') \Rightarrow \forall \sigma \in \Psi : \neg\hat{\mathcal{P}}(\tau'\sigma) \tag{2}$$

$$\forall \sigma \in \Psi : \neg\hat{\mathcal{P}}(\sigma) \Rightarrow \exists i : \neg\hat{\mathcal{P}}(\sigma[..i]) \tag{3}$$

$$\sigma \notin \Gamma \Rightarrow \exists i : (\forall \tau \in \Psi : \sigma[..i]\tau \notin \Gamma) \tag{4}$$

Based upon this, it can be shown that the class of security properties enforced by the monitor correspond to the class of co-recursively countable properties of programs, thereby circumscribing the class Π_1 of the arithmetic hierarchy.

Rewriting properties. The third class of properties in this hierarchy are those which can be enforced with the modification of the program. The definition of rewriting requires an adequate notion of equivalence. (See Sect. 4 for details.) For a given equivalence relation $\approx$, a rewriting mechanism R is considered "adequate" for enforcement if it exhibits the following properties:

$$P(R(M)) \tag{5}$$

$$P(M) \Rightarrow M \approx R(M) \tag{6}$$

These properties express that: Eq. (5) the modified program must guarantee the compliance with a policy; and Eq. (6) if a program complies with a policy before the rewriting, then it also complies with if after a modification. Hamlen et al. [16] show that there are properties which can solely be enforced with rewriting. However, in contrast to the previous classes, the rewriting mechanism does not corresponds to a class in the arithmetic hierarchy. That is, there is no known definition for a set of formal languages whose words denote properties, which can be enforced with rewriting. Conversely, it can be shown that the classes of properties Π_0 (static verification) and Π_1 (runtime monitoring) can be enforced with rewriting mechanisms.

3 Building Blocks: Processes Models and Policies

The previous section demonstrates that there is a class of properties which demand process rewriting for their enforcement. This section describes the technical building blocks that serve as input to ReWrite, namely process specifications and policies, as depicted in Fig. 1.

3.1 μBPMN Syntax

Our rewriting approach considers μBPMN as the modeling language for business processes. μBPMN is a Turing complete subset of BPMN containing its main constructs a formal semantics. Turing completeness is insofar relevant as it guarantees that all the computable processes can be modeled with and reasoned about in μBPMN. The formal semantics is essential to allow the rewriting and prove its correctness with regard to the congruence relations.

The μBPMN syntax possesses both a graphical and an algebraic notation. The graphic notation is the same as the corresponding of BPMN construct, whereas only the subset of the elements depicted in Fig. 5 is considered. The algebraic formalization is required to give semantics to the language, which is in this case based upon π-calculus [27]. (The latter is omitted in this paper.)

The graphical notation of μBPMN depicted in Fig. 5 consists of activities (subprocesses, loops and sub-process loops) and the gateways AND, OR and XOR, which allow for the complete representation of logical constructs in the control flow. Further, activities are linked with sequence arrows. Start and end markers complete the subset of BPMN used in this paper. Based upon [22], we have defined translations of μBPMN to BPEL – for execution – and workflow nets – for reasoning, e.g. soundness.

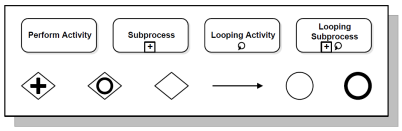


Fig. 5. μBPMN language.

The consideration of subset of BPMN does not restrict the applicability of the approach. First, because more complex modeling elements are seldom employed. According to [32], process models in industry usually consist of a few activities with complex branches. Second, because the language can be extended using the existing building blocks, should that be required.

Process modeled using the μBPMN are mapped a subset of the π-calculus. Processes are thus described by a set of activities that are triggered one after the other, provided the preconditions for their firing hold.

Relevant bits of μBPMN semantics. μBPMN has an operational semantics defined via an abstract process execution engine – mimicking the operation of existing BPMN engines – whose function is to determine the state transitions (choice of the next activity) and the execution environment for the process execution (provision of runtime parameters).

For the purposes of this paper, it is enough to consider a trace-based semantics for μBPMN based upon this calculus. Let Π be a process, a path χ of Π is defined as a sequence of input/output operations of Π upon the input $\omega \in \Sigma$. The set of all possible input parameters of a process Π is denoted by Γ^{ω}; $\Pi(\omega) \rightarrow \chi_{\omega}$ denotes the path triggered by inputting ω into Π, which eventually produces the path denoted by χ_{ω}^{Π}. Finally, Ξ_{Π} denotes the set of all paths in a process generated by $\Pi(\Gamma^{\omega})$. The congruence relations defined in Sect. 4 build upon reductions of generated paths in order to describe the changes (rewriting) in the process structure before the runtime. These changes are denoted as follows: $n \rightsquigarrow \chi_{\omega}$ when activity names are deleted in one path; $\mathcal{N} \rightsquigarrow \chi_{\omega}$ if activity names are deleted in all the possible paths of Π.

3.2 Compliance Rules

Figure 6 depicts the overall structure of a compliance rule in ReWrite. Each rule consists of a scope, a modality and a control. The *scope* defines the process and, therein, the control flow to which the rule applies. The *modality* describes how

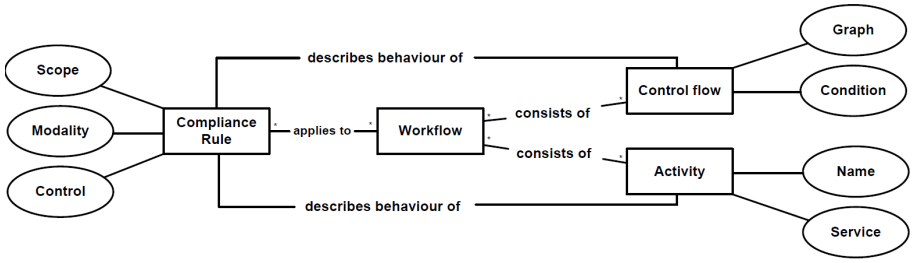


Fig. 6. The structure of compliance rules.

and where to apply the rule in a particular scope. The *control* defines which set of activities are to be triggered, as well as their order, so to ensure rule compliance.

This structure is expressive enough to capture the so-called "usage control" policies and, hence, the majority of regulatory compliance requirements on automated processes [24]. Generally, usage control policies refer to control flow constraints based upon the patterns in Dwyer et al. [13]. That is, they regard, for example, the absence, presence and cardinality of events, as well as their interdependencies (e.g. one event as a response of the other and mutual exclusion of event pairs). The evaluation envisaged for ReWrite considers these patterns.

More specifically, the class diagram depicted in Fig. 7 shows how the compliance rules are implemented in ReWrite. The scope consists of a list of processes to which the rule applies and one or more XPath expressions that describe the integration of (the controls specified in) this rule into the set of processes. The modality defines how the set of activities defined in the scope are treated. These activities can be deleted, replaced or complement with other activities. The latter (i.e. appending) may have different modes. For example, one can distinguish between appending before, after or during an activity (in one execution branch of a parallel execution). Similarly, one can add time constraints (e.g. "within three hours") and cardinality constraints (e.g. "exactly three times"). The controls with which the modalities are added to the process (process rewriting) are described as fragments of the BPEL language.

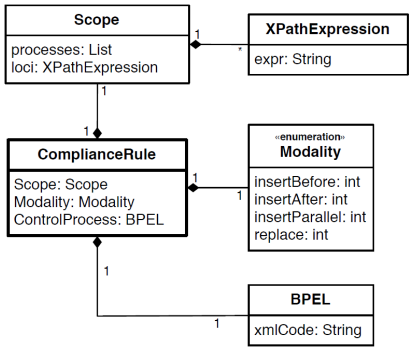


Fig. 7. Class diagram for rules.

We define a XML schema in order to express policies rules in a machine readable format. Figure 8 depicts a policy specified in this XML schema. This rule applies to two processes (`bpPatientReception` and `bpPatientInformation`), in particular their parts (`loci`-tag) `blood_pressure` and `weight` (here we replace the corresponding XPath expressions for simplicity). The `modality`-tag `insertBefore` conveys that the control process must be inserted before these `loci`. The control process, specified in the tag `BPELFragment`, can require in both

```xml
<?xml version="1.0" encoding="UTF-8"?>

<!--
    Document    : ComplianceRuleSchema.xml
    Author      : Sebastian Höhn
    Description :
        Sample document to test the schema validation
        for compliance rule schema.
-->

<ns0:complianceRule xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
    xmlns:ns0='http://xml.netbeans.org/schema/ComplianceRuleSchema'
    xsi:schemaLocation=
    'http://xml.netbeans.org/schema/ComplianceRuleSchema ComplianceRuleSchema.xsd'>
    <ns0:scope>
        <ns0:Processes>
            <ns0:Process processName="bpPatientReception"/>
            <ns0:Process processName="bpPatientInformation"/>
        </ns0:Processes>
        <ns0:Loci>
            <ns0:Locus expr="//assign/copy[from='blood_pressure']"></ns0:Locus>
            <ns0:Locus expr="//assign/copy[from='weight']"></ns0:Locus>
        </ns0:Loci>
    </ns0:scope>
    <ns0:modality>
        <ns0:instertBefore/>
    </ns0:modality>
    <ns0:controlProcess>
        <ns0:BPELFragment>Some BPEL here</ns0:BPELFragment>
    </ns0:controlProcess>

</ns0:complianceRule>
```

Fig. 8. Example of compliance rule.

cases the authentication of users using, for instance, their employee information. The concrete service invocation is omitted here.

4 Congruence Relations and Rewriting Operators

This section sketches the rewriting operators necessary to ensure compliance. While modifying processes models one must ensure that the functional characteristics of processes are maintained. Therefore, before defining the rewriting operators, this section sketches the congruence relations operators must fulfill.

4.1 Congruence Relations

Determining the semantic congruence of two arbitrary processes is a not decidable problem [18]. We define atomic operators and a stepwise approach to changing processes which turns out to be decidable and preserve the congruence relations. This section presents four congruence relations which are defined in terms of process paths. Taking two processes p_1 and p_2, the relations are:

– *Full semantic congruence:* all the paths generated by the process p_1 can also be generated by the process p_2. Formally: $\Xi_{p_1} = \Xi_{p_2}$. This relation is unsuitable for rewriting, as its own definition prohibits modifications.

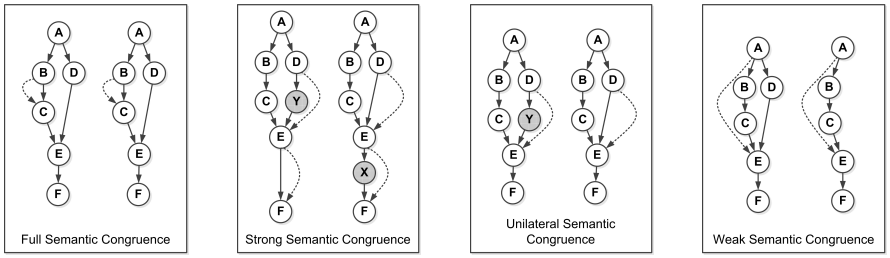


Fig. 9. Schematic illustration of the congruence relations.

- *Strong semantic congruence:* all the paths of the process p_1 can also be generated by the process p_2 after the reduction to common activity names. Formally: $\mathcal{N} = \Xi_{p_1} \cap \Xi_{p_2}$.
- *Unilateral semantic congruence:* the set of all paths generated by the process p_2 is a subset of the set of paths generated by p_1. Formally: $\Xi_{p_2} \subset \Xi_{p_1}$.
- *Weak semantic congruence:* At least one path generated the process p_1 is also a path of the process p_2. Formally: $\Xi_{p_1} \cap \Xi_{p_2} \neq \emptyset$.

Fig. 9 illustrates these relations. For simplicity, the processes are drawn as a simple state transition diagram (instead of μBPMN), with the dotted lines showing a possible move in the "congruence game". The shaded circles denote activities that have been added to the process flow. The pictures denote examples of a process flow that preserve the relations.

Lack of space prevents us from providing the formal definition of all these relations. Below we provide the formal definition for the strong semantic congruence, then jumping to the rewriting operators.

Definition 1. *Two processes p_1 and p_2 are strongly semantic congruent if the set of generated process paths after the reduction to the common names $\mathcal{N}_\mathcal{C} = \Xi_{p_1} \cap \Xi_{p_2} \neq \emptyset$ is identical: $(\mathcal{N}_\mathcal{C} \rightsquigarrow \Xi_{p_1}) = (\mathcal{N}_\mathcal{C} \rightsquigarrow \Xi_{p_2})$.* ⊣

It is easy to show that this relation is commutative. The strong semantic congruent relation allows changes in the process paths, as the activities names are reduced to a set of names common to both processes. In doing so, rewriting operators can be defined using this definition.

4.2 Rewriting Operators

The relations introduced in Sect. 4.2 are undecidable for two arbitrary processes. This follows from Jancar [18], who has proved that for Petri nets. His proof can be carry over to μBPMN: for each Petri net used in [18] we can build a corresponding μBPMN process, a fact that follows from the Turing completeness of both languages.

Still, for rewriting to work it is necessary to guarantee that the original and the rewritten processes are semantically congruent and that the latter indeed

	Append	Delete	Replace
Full	no	no	no
Strong	yes	yes	yes
Unilateral	partial	partial	no
Weak	partial	yes	partial

Table 1. Overview of the congruence guarantees for rewriting operators.

executes the controls required by the compliance policies. To achieve this in a decidable manner, we define atomic operators for appending, deleting and replacing activities in the process which, whenever applied in isolation or sequentially (one after the other), guarantee that the semantic congruence holds. Due to the lack of space, the following focuses on the "append" operator. An overview of the congruence guarantees for all the rewriting operators is given in Table 1.

The "append" operator. The "append" operator adds (missing or required) activities to the process model. This operator can be applied without disturbing the semantic congruence of processes. However, the following restrictions have to be made for the case of the unilateral and weak semantic congruence:

- If an activity has already been appended to one of the process models, then it is impossible to append further activities to the other model without disturbing the unilateral semantic congruence relation.
- If the two processes possess solely one common path, then appending one activity may disturb the congruence relations.

We have proved these properties for the corresponding congruence relations. For the strong semantic congruence, the following can be shown:

Theorem 1. *The append operator preserves the strong semantic congruence.* ⊣

Proof (Sketch). By appending an additional activity A to a process, then either Ξ_{P_1} or Ξ_{P_2} are extended with further traces. The computation of $\mathcal{N}_{\mathcal{C}} = \Xi_{P_1} \cap \Xi_{P_2}$ according to Def. 1 will, however, remove this extension (by renaming the activities), so that the processes still fulfill the strong semantic congruence. □

An analogous procedure can be used to demonstrate that the append operator preserves the unilateral and the weak semantic congruence relations, as shown in Table 1. Note that the "replace" operator is defined on the grounds of the primitive operators "append" and "delete".

Adequacy of semantic congruence. Establishing a relationship between the rewriting operators, congruence relations and the original business goals of a process is not trivial. Here, one can distinguish between the adequacy of different relations. Considering the strong semantic congruence and the "delete" operators, for example. It is possible to remove nearly all activities of a process and still fulfill the strong semantic congruence. Our experience shows that the

unilateral semantic congruence is the most promising relation within the ReWrite framework. It still suffers from the problem of the "delete" operator, but not to the same degree as the strong semantic congruence.

To tackle this problem, we add the following restrictions to the framework: (1) only activities that violate a compliance policy are deleted and, hence, should anyway *not* be executed; (2) if the semantic congruence is only violated at the same time that a compliance policy is violated, then the processes are still congruent. It can be shown that these restrictions are necessary and sufficient to tackle the problems arising from the "delete" operator.

5 Realization and Evaluation

The prototypical implementation of the ReWrite framework is based upon open-source technologies for the automation of processes with the standards of μBPMN, BPEL and automated translations from μBPMN into BPEL and the XML tools. This section reports on the realization of ReWrite and its envisaged evaluation.

The ReWrite framework has been designed as a component of the Security Workflow Analysis Toolkit (short, SWAT) [5]. SWAT is an Eclipse-based, extensible toolkit for the automated, well-founded security analysis of business processes models to analyze process models in a multitude of ways. SWAT provides for the following features: *process editing*, with import and export functions; *process simulation* to generate log files and configure policy violations using OpenESB as execution platform; *policy editing* to specify security and compliance policies; and *workflow analysis* to check whether process models comply with properties. See [5] for details.

Figure 10 depicts the architecture of ReWrite in SWAT. The design strategy while integrating ReWrite into SWAT is the "security-as-a-service" approach. Correspondingly, the architecture consists of two modules, namely: (1) *Modeling* for process design and policy specification; and (2) *Execution* for the rewriting of non-compliant process models and the automated execution of processes. Regarding (1), SWAT offers support for process modeling in μBPMN, BPEL and Petri nets, whereas for process execution the specifications are translated into BPEL. Compliance policies are specified using the XML editor "Oxygen", which is embedded in SWAT. Ongoing work is designing a policy editor and consistency checker for compliance rules. Regarding (2), SWAT employs OpenESB as an execution platform, whereas Glassfish acts an application server. It should be noted, though, that the actual realization

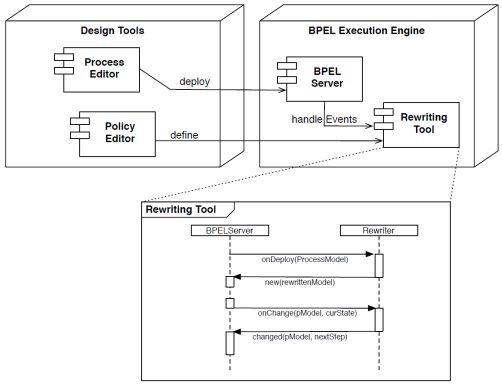


Fig. 10. ReWrite in SWAT.

of ReWrite does not require these technologies. Ongoing work is testing with a realization based upon jBPM.

The core of rewriting is actual modification of processes. The algorithms implementing these operators must guarantee, one the one hand, that the congruence relations are preserved (see Sect. 4.2) and, on the other, that the produced process models are correct with regard to the compliance policy. The latter depends on the compliance policy (i.e. controls defined therein) that are applied to the process. This is, however, out of the scope of this paper, as well as determining inconsistencies among policies.

The strategies for actually rewriting the processes is based on XSLT patterns, which are responsible for the transformation of XML documents (process models) with stylesheets. (Recall that the policy defines the scope of the rule, i.e., where to rewrite the process.) The following shows this using the append operator.

Example 2. To guarantee the integrity of data, one can trigger controls after each data input. Considering a health care setting, for example, one could demand that, after inputting the results of an exam, the patient ID should be entered again to avoid mistakes. The realization of this requirement demands inserting a control after one activity. The corresponding algorithm thus replaces one activity with a pair activity and control. The XSLT template to enforce this control is as follows (we omit the actual call to the service realizing the control):

```
<xsl:template match=''bpel:assign[@actionType='ControlAfterEach']''>
 <bpel:sequence name=''Rule_Verify''>
  <bpel:assign name=''EnterResult'' policy:actionType=''Result''>
    <xls:apply-template/>
  </bpel:assign>
  <bpel:assign name=''VerifyPatientID''>
    Replace by actual control from compliance rule
  </bpel:assign>
 </bpel:sequence>
</xsl:template>
```

This template employs the namespace `policy`, with which action types can be assigned to activities. This facilitates the rewriting in processes where multiple occurrences of the same activity happen and are in the scope of a rule. ⊣

ReWrite envisaged evaluation. We plan to carried out an extensive evaluation of ReWrite, where qualitative and quantitative issues were of interest: firstly, which policies can be rewritten; secondly, what are the performance figures obtained in doing so. This section reports on the former.

To evaluate the expressiveness of ReWrite we plan to employ *workflow patterns* [29] and *compliance policy patterns* [13]. Specifically, we take a representative subset workflow patterns as the minimal process specification. These patterns can be seen as appropriate building blocks for process specifications and, hence, if rewriting succeeds for these patterns, it also succeeds for more complex specifications composed using workflow patterns. The compliance rules build upon the patterns of Dwyer et al. [13]. These patterns characterize a representative set of primitive structural requirements of programs, such as the absence, precedence and bounded existence of activities. However, the patterns can equally well be applied to business processes [3, 25]. More importantly though,

the properties characterized by the patterns lie in the class of properties whose enforcement requires rewriting (see Sect. 2).

To actually assess the expressiveness of ReWrite, we need to determine which policy pattern can be added to which workflow pattern and, further, which could be alternatively enforced with an execution monitor (EM) and which demand rewriting (RW). Our goal is to demonstrate that each workflow pattern can be rewritten to comply with the corresponding policy pattern. Ongoing experiments deliver very promising preliminary results that substantiate this conjecture.

6 Related Work

Approaches to assessing business process compliance can be classified as [8, 25]: (1) *forward compliance checking* aims to design and implement processes where conforming behavior is enforced and (2) *backward compliance checking* aims to detect and localize non-conforming behavior. ReWrite is a forward compliance checking approach based on business process models and compliance policies. Related work addresses the annotation of business processes [15, 23], requirements elicitation [11, 10, 26] and formal verification [2–4, 6].

Program rewriting is a mechanism for enforcing security properties [16, 19]. It was initially formalized by Hamlen et al. [16] and he provided an implementation of a certified program-rewriting mechanism. These rewriting mechanisms are used to enforce low level properties such as type safety and do not consider business processes or high level concepts of modern programming languages. Similar approaches based on type systems to enforce certain security properties such as memory safety exist, among others, for Java bytecode [21], the .NET framework [14]. To our knowledge no approaches exist that investigate the transfer of automated rewriting techniques to business processes on a formal level.

The approaches discussed in the previous paragraph do not take into account the achievement of the programs' goals. The definition of some congruence relation is given, e.g. [19], but it is never explicitly specified in a way that it becomes possible to actually evaluate this relation for real world application.

De Backer and Snoeck discuss a concept called semantic compatibility which results in definitions of similar types of congruences [9]. As opposed to our approach, they are based on the languages defined by Petri Nets and they do not cover congruence of the processes themselves. They investigate how two processes that are deployed by different participants who need to achieve common business goals are able to cooperate. This notion of "compatibility" they devise is not applicable to the scenarios investigated in our research, because we discuss the business goals of a single process and not the interplay between two distinct processes in distinct administrative domains.

7 Summary and Further Work

This paper introduces a framework for rewriting business processes, thereby enforce security, compliance and privacy policies. Specifically, it motivates rewriting

by showing that existing enforcement mechanisms cannot cope with a relevant class of properties. The paper then defines the syntax and semantics for a graphical process modeling language and that of compliance rules. In order to ensure that the rewritten process still allows for the execution paths of the original process and, thereby, achieves the original business goals, different congruence relations are defined. Process rewriting must then not only correct the process, but also preserve some of these relations (depending on the kind of operation).

Lessons learned. Rewriting is an established field in the theory of programs and logics. This paper carries over some of these concepts into business process compliance management. Although rewriting is in general undecidable for chains of modifications, this paper shows that it is possible to define primitive operators that allow for decidable procedures. The ReWrite framework thus opens up the possibility of automated correction of processes to ensure process compliance "by design" (for modeled or reconstructed processes) or during runtime.

Further work. Besides the ongoing and future work already indicated in the text, future work comprises four directions: *firstly*, on the formal side it is still necessary to investigate the congruence relations. Spefically, we need to flesh out the details of the most sutiable relation to cover all the operators. *Secondly*, we see a relationship between the techniques we employ and process repositories. That in the sense that the same technologies for querying could be employed to support rewriting. Clarifying this interplay is subject of further work. *Thirdly*, the kinds of policy supported by ReWrite can be generalized to also consider security properties and other usage control policies. Future work will tackle the expressive power of policies and corresponding analysis techniques (e.g. inconsistency detection). Fourthly, testing ReWrite for monitoring process instances.

References

1. R. Accorsi. Sicherheit im prozessmanagement. digma *Zeitschrift für Datenrecht und Informationssicherheit*, 2013.
2. R. Accorsi and A. Lehmann. Automatic information flow analysis of business process models. In *BPM*, volume 7481 of *LNCS*, pages 172–187. Springer, 2012.
3. R. Accorsi, L. Lowis, and Y. Sato. Automated certification for compliant cloud-based business processes. *BISE*, 3(3):145–154, 2011.
4. R. Accorsi and C. Wonnemann. Strong non-leak guarantees for workflow models. In *ACM SAC*, pages 308–314. ACM, 2011.
5. R. Accorsi, C. Wonnemann, and S. Dochow. SWAT: A security workflow toolkit for reliably secure process-aware information systems. In *ARES*, pages 692–697. IEEE, 2011.
6. R. Agrawal, C. Johnson, J. Kiernan, and F. Leymann. Taming compliance with sarbanes-oxley internal controls using database technology. In *ICDE*, pages 92–101. IEEE, 2006.
7. B. Alpern and F. Schneider. Defining liveness. *IPL*, 21(4):181–185, October 1985.
8. A. Antón, E. Bertino, N. Li, and T. Yu. A roadmap for comprehensive online privacy policy management. *CACM*, 50(7):109–116, July 2007.
9. M. D. Backer and M. Snoeck. Business process verification: a petri net approach. Open access publications from Katholieke Universiteit Leuven, 2007.

10. T. Breaux and A. Antón. Analyzing regulatory rules for privacy and security requirements. *IEEE TSE*, 34(1):5–20, 2008.
11. T. Breaux, A. Antón, and E. Spafford. A distributed requirements management framework for legal compliance and accountability. *Computers & Security*, 28(1-2):8–17, 2009.
12. E. Y. Chang, Z. Manna, and A. Pnueli. Characterization of temporal property classes. In *ICALP*, volume 623 of *LNCS*, pages 474–486. Springer, 1992.
13. M. B. Dwyer, G. S. Avrunin, and J. C. Corbett. Patterns in property specifications for finite-state verification. In *IEEE CSE*, pages 411–420. ACM, 1999.
14. ECMA. Ecma-335: Common language infrastructure. european association for standardizing information and communication systems. Tech. Rep., ECMA, 2002.
15. A. Ghose and G. Koliadis. Auditing business process compliance. In *ICSOC*, volume 4749 of *LNCS*, pages 169–180. Springer, 2007.
16. K. Hamlen, G. Morrisett, and F. Schneider. Computability classes for enforcement mechanisms. *ACM TOPLAS*, 28(1):175–205, January 2006.
17. M. Hilty, D. Basin, and A. Pretschner. On obligations. In *ESORICS*, volume 3679 of *LNCS*, pages 98–117. Springer, 2005.
18. P. Jancar. Undecidability of bisimilarity for petri nets and some related problems. *Theor. Comput. Sci.*, 148(2):281–301, 1995.
19. R. Khoury and N. Tawbi. Corrective enforcement: A new paradigm of security policy enforcement by monitors. *ACM TISSEC*, 15(2):10, 2012.
20. M. Leucker and C. Schallhart. A brief account of runtime verification. *J. Logic and Algebraic Programming*, 78(5):293–303, May/June 2008.
21. T. Lindholm and F. Yelling. *The Java Virtual Machine*. Addison-Wesley, 1999.
22. N. Lohmann, E. Verbeek, and R. Dijkman. Petri net transformations for business processes - A survey. In *TPNOMC*, volume 5460 of *LNCS*, pages 46–63. Springer, 2009.
23. K. Namiri and N. Stojanovic. Using control patterns in business processes compliance. In *WISE*, volume 4832 of *LNCS*, pages 178–190. Springer, 2007.
24. A. Pretschner, F. Massacci, and M. Hilty. Usage control in service-oriented architectures. In *TRUSTBUS*, volume 4657 of *LNCS*, pages 83–93. Springer, 2007.
25. E. Ramezani, D. Fahland, and W. M. P. van der Aalst. Where did i misbehave? diagnostic information in compliance checking. In *BPM*, volume 7481 of *LNCS*, pages 262–278. Springer, 2012.
26. S. Sadiq, G. Governatori, and K. Namiri. Modeling control objectives for business process compliance. In *BPM*, volume 4714 of *LNCS*, pages 149–164. Springer, 2007.
27. D. Sangiorgi and D. Walker. *The pi-Calculus: A Theory of Mobile Processes*. Cambridge Press, 2001.
28. F. Schneider. Enforceable security policies. *ACM TISSEC*, 3(1):30–50, 2000.
29. W. van der Aalst. Workflow patterns. In *Encyclopedia of Database Systems*, pages 3557–3558. Springer, 2009.
30. W. van der Aalst. *Process Mining – Discovery, Conformance and Enhancement of Business Processes*. Springer, 2011.
31. M. Weske. *Business Process Management: Concepts, Languages and Architectures*. Springer, 2011.
32. C. Wolter, M. Menzel, A. Schaad, P. Miseldine, and C. Meinel. Model-driven business process security requirement specification. *J. Systems Architecture*, 55(4):211–223, 2009.