Minseok Song
Moe Thandar Wynn
Jianxun Liu (Eds.)

# Asia Pacific Business Process Management

LNBIP 159

First Asia Pacific Conference, AP-BPM 2013
Beijing, China, August 2013
Selected Papers

Lecture Notes
in Business Information Processing 159

Series Editors

Minseok Song
Moe Thandar Wynn
Jianxun Liu (Eds.)

# Asia Pacific Business Process Management

First Asia Pacific Conference, AP-BPM 2013
Beijing, China, August 29-30, 2013
Selected Papers

Springer

Volume Editors

Minseok Song
Ulsan National Institute of Science and Technology
School of Technolgy Management
Ulsan, South Korea
E-mail: msong@unist.ac.kr

Moe Thandar Wynn
Queensland University of Technology
Information Systems School
Brisbane, OLD, Australia
E-mail: m.wynn@qut.edu.au

Jianxun Liu
Hunan University of Science and Technology
School of Computer Science and Engineering
Xiangtan, China
E-mail: ljx529@gmail.com

ISSN 1865-1348 e-ISSN 1865-1356
ISBN 978-3-319-02921-4 e-ISBN 978-3-319-02922-1
DOI 10.1007/978-3-319-02922-1
Springer Cham Heidelberg NewYork Dordrecht London

Library of Congress Control Number: 2013951177



*Typesetting:* Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

*The original version of the book was revised: The copyright line was incorrect. The Erratum to the book is available at DOI: 10.1007/978-3-319-02922-1_10*

# Preface

These proceedings contain the final versions of papers for the inaugural Asia-Pacific Conference on Business Process Management (AP-BPM) 2013 that took place in Beijing, China. AP-BPM 2013 was the first edition of the conference for researchers and practitioners in the field of business process management (BPM) in the region. Its purpose is to provide a high-quality forum for researchers and practitioners to exchange research findings and ideas on BPM technologies and practices that are highly relevant to the Asia-Pacific region. Through this conference, we aim to set up a bridge between actual industrial requirements and leading-edge research outcomes for the growth of the economic rising powers of the Asia-Pacific region. More information on AP-BPM conference can be found on the website: http://www.ap-bpm.org

This year we had 19 submissions from seven countries (Australia, China, Guatemala, Indonesia, Italy, South Korea, Taiwan). Following an extensive review process by an international Program Committee, eight papers (seven full papers and one short paper) were accepted for publication in these proceedings and presentation at the conference. In addition to the regular presentations, a keynote speech was delivered by Prof. Wil van der Aalst and the keynote paper is also included in the proceedings.

We would like to thank the authors for their submissions to the conference and Prof. Wil van der Aalst for his excellent keynote. We also would like to express our gratitude to Program Committee members and paper reviewers, and acknowledge the support of the Steering Committee. Finally, we would like to thank Lijie Wen for chairing the Organizing Committee.

August 2013

Minseok Song
Moe Thandar Wynn
Jianxun Liu

# Conference Organization

## Honorary Chair

| | |
|---|---|
| Jiaguang Sun | Tsinghua University, China |

## General Chair

| | |
|---|---|
| Jianmin Wang | Tsinghua University, China |

## International Advisory Committee

| | |
|---|---|
| Arthur ter Hofstede | Queensland University of Technology, Australia |
| Hyerim Bae | Pusan National University, Korea |
| Jianmin Wang | Tsinghua University, China |
| Pingyu Hsu | National Central University, Taiwan |
| Budi Santosa | ITS, Indonesia |

## Steering Committee

### Honorary Chair

| | |
|---|---|
| Arthur ter Hofstede | Queensland University of Technology, Australia |

### Executive Chair

| | |
|---|---|
| Hyerim Bae | Pusan National University, Korea |
| Jianmin Wang | Tsinghua University, China |

## Program Chairs

| | |
|---|---|
| Minseok Song | Ulsan National Institute of Science and Technology, Korea |
| Moe Thandar Wynn | Queensland University of Technology, Australia |
| Jianxun Liu | Hunan University of Science and Technology, China |

## Organization Chair

| | |
|---|---|
| Lijie Wen | Tsinghua University, China |

### Publicity Chair

| | |
|---|---|
| Yahui Lu | Shenzhen University, China |

## Program Committee

| | |
|---|---|
| Saiful Akbar | ITB, Indonesia |
| Yudistira Dwi Wardhana Asnar | ITB, Indonesia |
| Joonsoo Bae | Chonbuk National University, Korea |
| Jian Cao | Shanghai Jiao Tong University, China |
| Namwook Cho | Seoul National Univ. of Technology, Korea |
| Lizhen Cui | Shandong University, China |
| ZaiwenFeng | Wuhan University, China |
| Yingbo Liu | Tsinghua University, China |
| Yahui Lu | Shenzhen University, China |
| Xiangpei Hu | Dalian University of Technology, China |
| Pingyu Hsu | National Central University, Taiwan |
| Jae-yoon Jung | Kyunghee University, Korea |
| Dongsoo Kim | Soongsil University, Korea |
| Kwanghoon Kim | Kyonggi University, Korea |
| Minsoo Kim | Pukyung National University, Korea |
| Michiharu Kudo | IBM Research, Japan |
| Anto Satriyo Nugroho | Center for Information & Communication Technology, Indonesia |
| Chao Ou-Yang | National Taiwan University of Science and Technology, Taiwan |
| Chun Ouyang | Queensland University of Technology, Australia |
| Punnamee Sachakamol | Kasetsart University, Thailand |
| Shazia Sadiq | University of Queensland, Australia |
| Budi Santosa | ITS, Indonesia |
| Riyanarto Sarno | ITS, Indonesia |
| Ricardo Seguel | BPM LATAM, Chile |
| Markus Stumptner | University of South Australia, Australia |
| Pablo David Villarreal | National Technological University, Argentina |
| Xuefeng Wang | Harbin Institute of Technology, China |
| Zhaoxia Wang | Logistical Engineering University, China |
| Lijie Wen | Tsinghua University, China |
| Yiping Wen | Central South University, China |
| Raymond Wong | The University of New South Wales, Australia |
| Jei-Zheng Wu | Soochow University, Taiwan |
| Bernardo N. Yahya | UNIST, Korea |
| Yang Yu | Sun Yat-sen University, China |
| Chongyi Yuan | Peking University, China |
| Haiping Zha | Naval Armament Research Institute of PLA, China |
| Li Zhang | Tsinghua University, China |
| Liang Zhang | Fudan University, China |

| | |
|---|---|
| Yang Zhang | Beijing University of Posts and Telecommunications, China |
| Lindu Zhao | Southeast University, China |
| Wen Zhao | Peking University, China |

# Table of Contents

# Process Cubes: Slicing, Dicing, Rolling Up and Drilling Down Event Data for Process Mining

Wil M.P. van der Aalst

Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, The Netherlands
Business Process Management Discipline, Queensland University of Technology, Brisbane, Australia and
International Laboratory of Process-Aware Information Systems, National Research University Higher School of Economics, Moscow, Russia
w.m.p.v.d.aalst@tue.nl

**Abstract.** Recent breakthroughs in *process mining* research make it possible to discover, analyze, and improve business processes based on event data. The growth of event data provides many opportunities but also imposes new challenges. Process mining is typically done for an isolated well-defined process in steady-state. However, the boundaries of a process may be fluid and there is a need to continuously view event data from different angles. This paper proposes the notion of *process cubes* where events and process models are organized using different dimensions. Each cell in the process cube corresponds to a set of events and can be used to discover a process model, to check conformance with respect to some process model, or to discover bottlenecks. The idea is related to the well-known OLAP (Online Analytical Processing) data cubes and associated operations such as slice, dice, roll-up, and drill-down. However, there are also significant differences because of the process-related nature of event data. For example, process discovery based on events is incomparable to computing the average or sum over a set of numerical values. Moreover, dimensions related to process instances (e.g. cases are split into gold and silver customers), subprocesses (e.g. acquisition versus delivery), organizational entities (e.g. backoffice versus frontoffice), and time (e.g., 2010, 2011, 2012, and 2013) are semantically different and it is challenging to slice, dice, roll-up, and drill-down process mining results efficiently.

**Keywords:** OLAP, Process Mining, Big Data, Process Discovery, Conformance Checking.

## 1 Introduction

Like most IT-related phenomena, also the growth of event data complies with Moore's Law. Similar to the number of transistors on chips, the capacity of hard disks, and the computing power of computers, the digital universe is growing exponentially and roughly doubling every 2 years [35, 40]. Although this is not a

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-319-02922-1_10

M. Song, M.T. Wynn, and J. Liu (Eds.): AP-BPM 2013, LNBIP 159, pp. 1–22, 2013.

new phenomenon, suddenly many organizations realize that increasing amounts of "Big Data" (in the broadest sense of the word) need to be used intelligently in order to compete with other organizations in terms of efficiency, speed and service. However, the goal is not to collect as much data as possible. The real challenge is to turn event data into valuable insights. Only *process mining* techniques directly relate event data to end-to-end business processes [1]. Existing business process modeling approaches generating piles of process models are typically disconnected from the real processes and information systems. Data-oriented analysis techniques (e.g., data mining and machines learning) typically focus on simple classification, clustering, regression, or rule-learning problems.

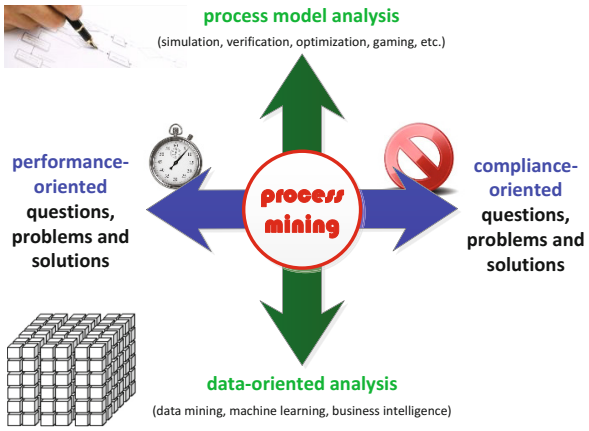



**Fig. 1.** Process mining provides the missing link between on the one hand process model analysis and data-oriented analysis and on the other hand performance and conformance

Process mining aims to *discover, monitor and improve real processes by extracting knowledge from event logs* readily available in today's information systems [1]. Starting point for any process mining task is an *event log*. Each event in such a log refers to an *activity* (i.e., a well-defined step in some process) and is related to a particular *case* (i.e., a *process instance*). The events belonging to a case are *ordered* and can be seen as one "run" of the process, i.e., an event log can be viewed as a collection of *traces*. It is important to note that an event log contains only *example behavior*, i.e., we cannot assume that all possible traces have been observed [1].

The growing interest in process mining is illustrated by the *Process Mining Manifesto* [36] recently released by the *IEEE Task Force on Process Mining*. This manifesto is supported by 53 organizations and 77 process mining experts contributed to it.

The process mining spectrum is quite broad and includes techniques for process discovery, conformance checking, model repair, role discovery, bottleneck

analysis, predicting the remaining flow time, and recommending next steps. Over the last decade hundreds of process mining techniques have been proposed. A process discovery technique uses as input an event log consisting of a collection of traces (i.e., sequences of events) and constructs a process model (Petri net, BPMN model, or similar) that "adequately" describes the observed behavior. A conformance checking technique uses as input an event log and a process model, and subsequently diagnoses differences between the observed behavior (i.e., traces in the event log) and the modeled behavior (i.e., possible runs of the model). Different process model notations can be used, e.g., BPMN models, BPEL specifications, UML activity diagrams, Statecharts, C-nets, or heuristic nets. MXML or XES (`www.xes-standard.org`) are two typical formats for storing event logs ready for process mining.

The incredible growth of event data poses new challenges [53]. As event logs grow, process mining techniques need to become more efficient and highly scalable. Dozens of process discovery [1, 11, 12, 16, 30, 18, 24, 25, 28, 31, 41, 54, 60, 61] and conformance checking [6, 13, 14, 15, 22, 29, 31, 42, 43, 51, 59] approaches have been proposed in literature. Despite the growing maturity of these approaches, the quality and efficiency of existing techniques leave much to be desired. State-of-the-art techniques still have problems dealing with large and/or complex event logs and process models.

Whereas traditional process mining techniques focus on the offline analysis of solitary processes in steady-state, this paper focuses on multiple inter-related processes that may change over time. Processes may change due to seasonal influences, working patterns, new laws, weather, and economic development. Moreover, there may be multiple variants of the same process or the process is composed of subprocesses. Existing techniques also cannot handle multiple process variants and/or heterogeneous collections of cases. However, in reality the same process may be used to handle very different cases, e.g., in a care process there may be characteristic groups of patients that need to be distinguished from one another. Moreover, there may be different variants of the same process, e.g., different hospitals execute similar care processes, and it is interesting to compare them. Obviously, it is very challenging to discover and compare processes for different hospitals and patient groups. Unfortunately, traditional techniques tend to focus on a single well-defined process. Cases can be clustered in groups and process models can be compared, however, *there are no process discovery techniques that produce overarching models able to relate and analyze different groups and process variants*. For example, we have applied process discovery in over 25 municipalities executing similar processes. However, there are no discovery approaches relating these process variants.

In this paper, we propose the new notion of *process cubes* where events and process models are organized using different dimensions (e.g., case types, event classes, and time windows). A *process cube* may have any number of dimensions used to distribute process models and event logs over multiple cells. The first process cube shown in Figure 2(left) has three dimensions: *case type*, *event class* and *time window*. In this simple example, there is only one case type and only

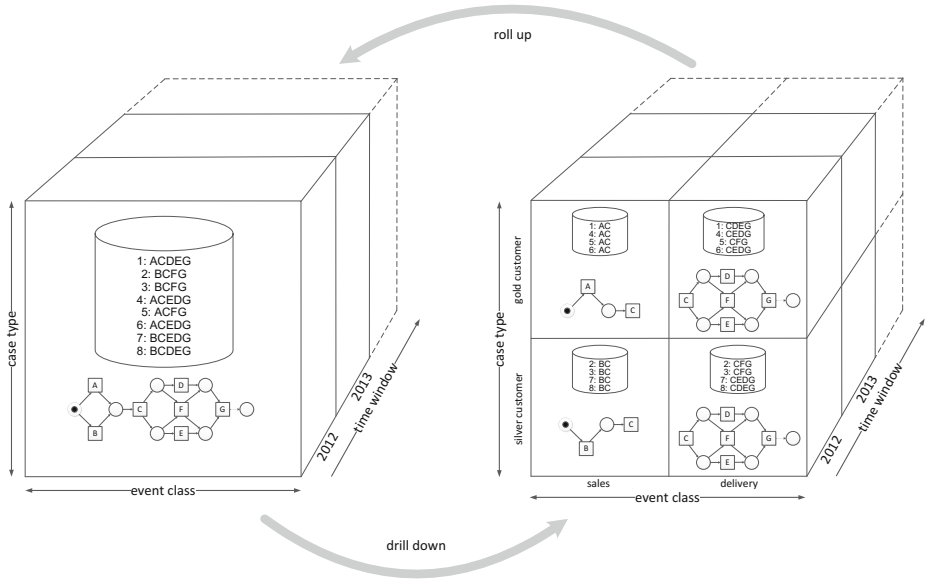



**Fig. 2.** Two process cubes illustrating the splitting (drilling down) and merging (rolling up) of process cells using the case type and event class dimensions

one event class. The cube covers multiple time windows, but only one is shown (all cases completed in 2012). In this toy example there are only eight cases (i.e., process instances) and seven distinct activities. The process may be split by identifying multiple case types and/or multiple event classes. The second process cube shown on the right-hand side of Figure 2 has two case types (gold customer and silver customer) and two event classes (sales and delivery).

The *case type dimension* is based on properties of the case the event belongs to. In Figure 2(right), cases 1, 4, 5, and 6 refer to a "gold customer". Hence, the cells in the "gold customer" row include events related to these four cases. The *event class dimension* is based on properties of individual events, e.g., the event's activity name, its associated resource, or the geographic location associated with the event. In Figure 2(right), the event class dimension is based on the activity of each event. The event class "sales" includes activities $A$, $B$, and $C$. The event class "delivery" refers to activities $C$, $D$, $E$, $F$, and $G$. The *time window dimension* uses the timestamps found in the event log. A time window may refer to a particular day, week, month, or any other period.

*Each cell in a process cube refers to a collection of events* and possibly also process mining results (e.g., a discovered process model) or other artifacts (e.g., a set of business rules). Events may be included in multiple cells, e.g., sales and delivery cells share $C$ events. Each of the three dimensions may have an associated hierarchy, e.g., years composed of months and months composed of days.

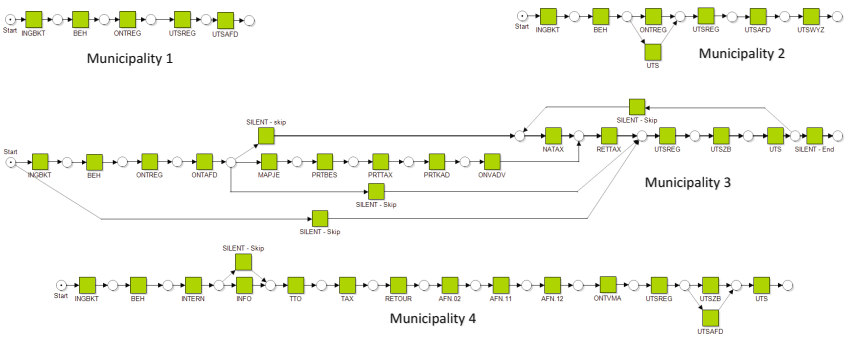



**Fig. 3.** Process models showing how complaints regarding the taxation of real estate are handled within four Dutch municipalities

Figure 3 illustrates the relevance of process cubes using four variants of the same municipal complaints handling process. The process models in Figure 3 show that four of the ten municipalities involved in our CoSeloG project[1] are handling complaints related to the taxation of houses very differently [20]. For each of the four processes we have event data and normative process models. The average throughput times of the four process variants are very different, e.g., Municipality 1 handles complaints in 22 days whereas Municipality 3 uses 227 days. We aim at organizing such event data and process models in a process cube that allows for easy comparison of processes between municipalities, over time, and for different groups of citizens.

Process cubes are related to the well-known OLAP (Online Analytical Processing) cubes [27] and large process model repositories [49]. In an OLAP cube, one can drill-down or roll-up data, zoom into a selected slice of the overall data, or reorder the dimensions. However, OLAP cubes cannot be used for process-related data since events are ordered and belong to cases. Moreover, cells are associated to process models and not just event data. Conversely, process model repositories do not store event data. In process cubes, models and event data are directly related. Observed and modeled behavior can be compared, models can be discovered from event data, and event data can be used the breathe life into otherwise static process models.

This paper defines OLAP notions such as "slicing", "dicing", "rolling up" and "drilling down" for event data. These can be used to compare, merge, and split process cells at both the log and model level. The process cube notion is closely related to *divide-and-conquer approaches in process mining* where huge event logs are partitioned into smaller sublogs to improve performance and scalability. In principle, process cubes can also be used to decompose challenging process mining problems into smaller problems using the techniques described in [3, 5, 4]. These techniques may be used to speed-up OLAP operations.

[1] See the CoSeLoG (Configurable Services for Local Governments) project home page, `www.win.tue.nl/coselog`

The remainder of this paper is organized as follows. Section 2 introduces the *process cube* notion and further motivates its relevance. Section 3 formalizes the *event base* used to create process cubes, i.e., the source information describing "raw" events and their properties. The so-called *process cube structure* is defined in Section 4. This structure defines the *dimensions* of the cube. Event base and process cube structure are linked through the so-called *process cube view* defined in Section 5. Section 6 defines the *slice* and *dice* operations on process cubes. *Roll-up* and *drill-down* are formalized in Section 7. Section 8 concludes the paper by discussing innovations and challenges.

## 2 Process Cubes

As illustrated by Figure 4, event data can be used to construct a *process cube*. Each *cell* in the process cube corresponds to a *set of events* selected based on the corresponding *dimension* values. In Figure 4 there are three dimensions. However, a process cube can have any number of dimensions $n \in \mathbb{N}$. Moreover, dimensions can be based on any event property. In Figure 4 events are grouped in cells based on *case type*, a particular *event class*, and a particular *time window*, i.e., one cell refers to the set of all events belonging to case type $ct$, event class $ec$, and time window $tw$. The *case type dimension* is based on properties of the case as a whole and not on the characteristics of individual events. Hence, if event $e$ is of type $ct$, then all events of the case to which $e$ belongs, also have type $ct$. Case type $ct$ may be based on the type of customer (gold of silver) or on the total amount (e.g., $< 1000$ or $\geq 1000$). The *event class dimension* is based on properties of the individual events, e.g., the event's activity name, associated resources, or geographic location. Event type ($et$) may depend on the activity name, e.g., there could be three event classes based on overlapping sets of activity names: $\{A, B\}$, $\{C, D\}$, and $\{E\}$. The *time window dimension* uses the timestamps found in the event log. A time window ($tw$) may refer to a particular day, week, month, or any other period, e.g., to all events that took place in December 2012.

An event may belong to multiple process cells because case types, event classes, and time windows may be overlapping. Process cells may be merged into larger cells, i.e., event data can be grouped at different levels of granularity. Semantically, the merging of cells corresponds to the merging of the corresponding event sets. One may refine or coarsen a dimension.

A *process cube* is composed of a set of process cells as shown in Figure 4. Per cell one may have a predefined or discovered process model. The process model may have been discovered from the cell's event data or given upfront. Moreover, other artifacts, e.g., organizational models [56], may be associated to individual cells.

Process cubes will be used to relate different processes, e.g., we may be interested in understanding the differences between gold and silver customers, large orders and small orders, December and January, John and Ann, etc.

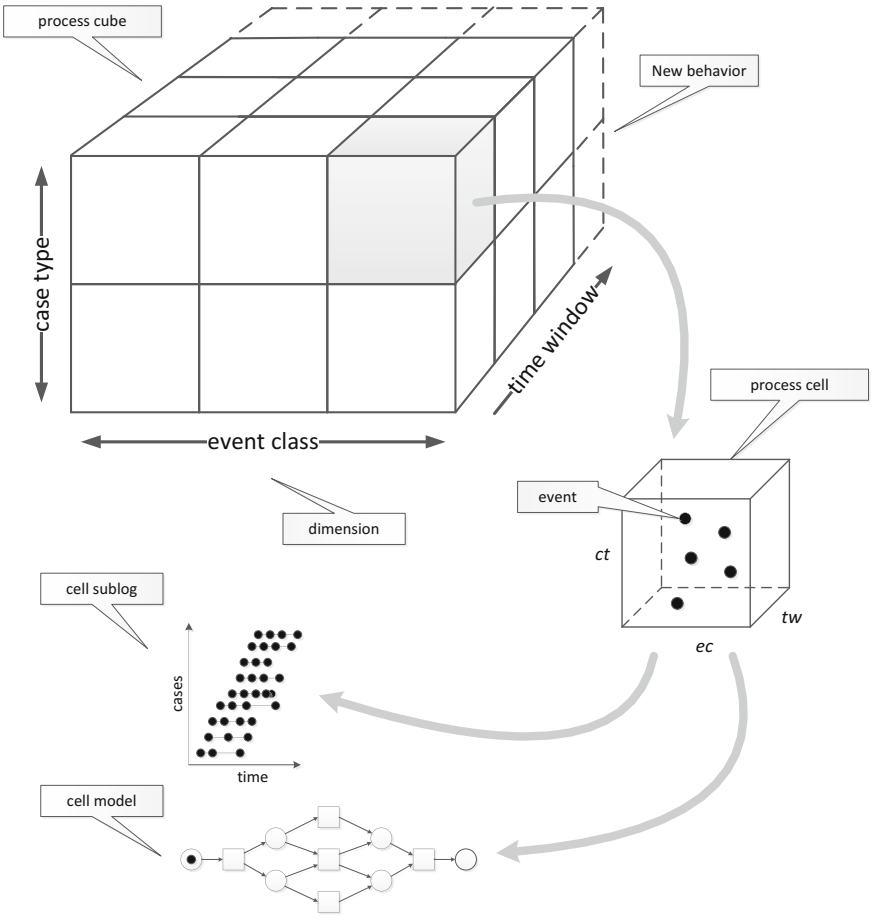



**Fig. 4.** A process cube relates events to different dimensions. Each cell in the cube corresponds to a sublog containing events and may have an associated set of models or other artifacts (derived or given as input).

Moreover, we may want to chop a larger cell into many smaller cells for efficiency reasons (e.g., distributing a time-consuming discovery task).

The three dimensions shown in Figure 4 only serve as examples and may be refined further, e.g., there may be multiple dimensions based on various classifications of cases (e.g., customer type, region, size, etc.). Moreover, each dimension may have a natural hierarchical structure (e.g., a year is composed of months and a country is composed of regions) that can be exploited for the aggregation, refinement, and selection of event data.

Process cells (and the associated sublogs and models) can be split and merged in two ways as is illustrated in Figure 5. The *horizontal dimension* of a cell refers to model elements (typically activities) rather than cases. The *vertical dimension* of a cell refers to cases rather than model elements. Consider for

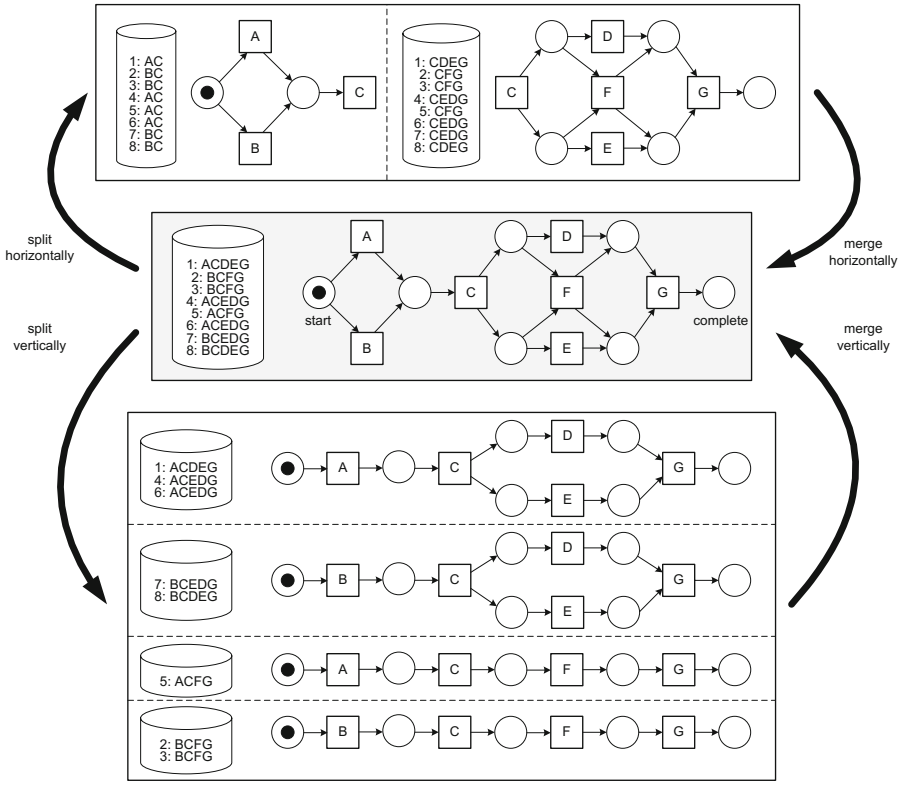



**Fig. 5.** Illustration of the two types of merging and splitting process cells

example the event log depicted in the middle of Figure 5. The event log consists of 8 cases and 37 events. If the log is split horizontally based on the two overlapping event classes $\{A, B, C\}$ and $\{C, D, E, F, G\}$, then we obtain two sublogs each consisting of all 8 cases. The top-left part of Figure 5 shows the new process cell corresponding to event class $\{A, B, C\}$. Model and event log abstract from activities $\{D, E, F, G\}$. The top-right part of Figure 5 shows the process cell corresponding to event class $\{C, D, E, F, G\}$. Note that the cell's sublog and model abstract from activities $A$ and $B$. If the log is split vertically, we could obtain the four sublogs depicted in the lower half of Figure 5. Each cell contains a subset of cases selected according to some criterion, e.g., the type of customer or the set of activities executed. Unlike the horizontal split, no individual events are removed, i.e., all events belonging to a case are included in the cell (or no events of the case are included).

The seven process models shown in Figure 5 may correspond to discovered or modeled behaviors. The models in the horizontally split cells are in a "part of" relationship, i.e., they are fragments of the larger model and cover only subsets of activities. The models in the vertically split cells are in an "is a" relationship, i.e., they can be viewed as specializations of original model covering only subsets

of cases. *The case type and time window dimensions in Figure 4 are often used to merge or split a log vertically. The event class dimension is often used to merge or split a log horizontally.*

Obviously, there are some similarities between a process cube and an OLAP (Online Analytical Processing) cube [27]. In an OLAP cube, one can drill-down or roll-up data, zoom into a selected slice of the overall data, or reorder the dimensions. As shown in [46], these ideas can be applied to event data. Any selection of cells in the process cube can be used to materialize an event log and discover the corresponding process model. However, unlike [46] which focuses on a multi-dimensional variant of the well-know heuristic miner [60], we aim at a much more general approach. On the one hand, we allow for any number of dimensions and any process mining technique (not just discovery). On the other hand, we take into account the essential properties of event data and processes. For example, the two types of merging and splitting process cells illustrated by Figure 5 are process-specific and do not correspond to existing OLAP notions. Techniques for merging and splitting process cells are related to divide-and-conquer approaches for process mining (e.g., to distribute process discovery or conformance checking) [3].

Based on the ideas shown in Figure 4, we have developed an initial prototype (called *ProCube*) using the process mining framework *ProM* and the *Palo OLAP* toolset (*JPalo* client and *Palo* MOLAP server) [39]. *ProCube* application runs as a plugin in *ProM*. *Palo* is employed for its OLAP capabilities. The *ProCube* plug-in creates sublogs per cell on-the-fly and visualizes process models discovered using the fuzzy [34] and heuristics [60] miner, social networks derived using ProM's social network miner [10], and dotted charts [55] computed per cell.

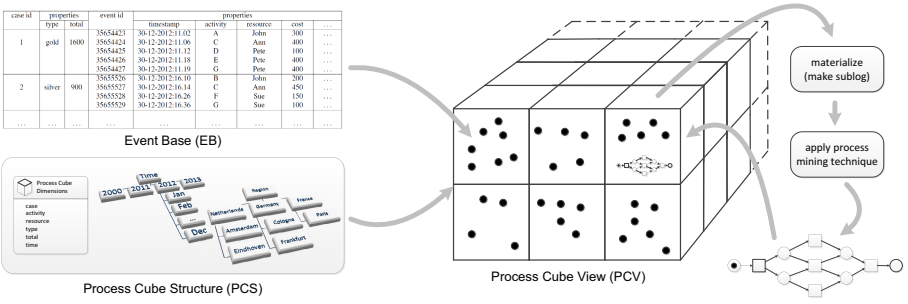



**Fig. 6.** Overview of the different ingredients needed to define and use process cubes

In the remainder, we do not focus on specific process mining techniques or a specific implementation of the process cube notion. Instead, we conceptualize the essential ideas. Figure 6 lists the different ingredients described next. The *Event Base* (EB) contains information about actually recorded events (Section 3). These events may have different properties, some of which are used as dimensions in the *Process Cube Structure* (PCS) described in Section 4.

**Table 1.** A fragment of some event log: each line corresponds to an event

| case id | properties | | event id | properties | | | |
|---|---|---|---|---|---|---|---|
| | type | total | | timestamp | activity | resource | cost |
| 1 | gold | 1600 | 35654423 | 30-12-2012:11.02 | A | John | 300 |
| | | | 35654424 | 30-12-2012:11.06 | C | Ann | 400 |
| | | | 35654425 | 30-12-2012:11.12 | D | Pete | 100 |
| | | | 35654426 | 30-12-2012:11.18 | E | Pete | 400 |
| | | | 35654427 | 30-12-2012:11.19 | G | Pete | 400 |
| 2 | silver | 900 | 35655526 | 30-12-2012:16.10 | B | John | 200 |
| | | | 35655527 | 30-12-2012:16.14 | C | Ann | 450 |
| | | | 35655528 | 30-12-2012:16.26 | F | Sue | 150 |
| | | | 35655529 | 30-12-2012:16.36 | G | Sue | 100 |
| ... | ... | ... | ... | ... | ... | ... | ... |

A *Process Cube View* (PCV) uses both EB and PCS to create a concrete view. The view may be modified using typical OLAP operations such as slice and dice (Section 6) and roll-up and drill-down (Section 7). Any process mining technique can be applied to a cell in the selected view. To do this, the cell's event data need to be materialized to create a sublog that is used as input by conventional process mining techniques. These techniques may produce (process) models, charts, etc. The results are be stored per cell and the different cells can be compared systematically.

## 3 Event Base

Normally, *event logs* serve as the starting point for process mining. These logs are created having a particular process and a set of questions in mind. An event log can be viewed as a multiset of *traces*. Each trace describes the life-cycle of a particular *case* (i.e., a *process instance*) in terms of the *activities* executed. Often event logs store additional information about events. For example, many process mining techniques use extra information such as the *resource* (i.e., person or device) executing or initiating the activity, the *timestamp* of the event, or *data elements* recorded with the event (e.g., the size of an order). Table 1 shows a small fragment of some larger event log. Only two traces are shown. Each event has a unique id and several properties. For example, event 35654423 is an instance of activity $A$ that occurred on December 30th at 11.02, was executed by John, and costs 300 euros. The second trace starts with event 35655526 and also refers to an instance of activity $A$. Note that each trace corresponds to a *case*, i.e., a completed process instance. Also cases may have properties as is shown in Table 1 where cases have a customer *type* (gold or silver) and total *amount*, e.g., case 1 was executed for a gold customer and had a total amount of 1600 euro. Implicitly, an event inherits the properties of the corresponding case.

For process cubes we consider an *event base*, i.e., a large collection of events not tailored towards a particular process or predefined set of questions. An event

base can be seen as an all-encompassing event log or the union of a collection of related event logs.

Properties of events have values and the dimensions of a process cube structure sets of possible property values. Throughout the paper we assume the following universes.

**Definition 1 (Universes).** *$\mathcal{U}_V$ is the universe of possible attribute values (e.g., strings, numbers, etc.). $\mathcal{U}_S = \mathcal{P}(\mathcal{U}_V)$ is the universe of value sets. $\mathcal{U}_H = \mathcal{P}(\mathcal{U}_S)$ is the universe of value set collections (set of sets).*

Note that $v \in \mathcal{U}_V$ is a single value (e.g., $v = 300$), $V \in \mathcal{U}_S$ is a set of values (e.g., $V = \{gold, silver\}$), and $H \in \mathcal{U}_H$ is a collection of sets. For example, $H = \{\{a, b, c\}, \{c, d\}, \{d, e, f\}\}$ or $H = \{\{x \in \mathbb{N} \mid x < 50\}, \{x \in \mathbb{N} \mid 50 \leq x < 60\}, \{x \in \mathbb{N} \mid x \geq 60\}\}$.

**Definition 2 (Event Base).** *An* event base *$EB = (E, P, \pi)$ defines a set of* events *$E$, a set of* event properties *$P$, and a function $\pi \in P \rightarrow (E \not\rightarrow \mathcal{U}_V)$. For any property $p \in P$, $\pi(p)$ (denoted $\pi_p$) is a partial function mapping events onto values. If $\pi_p(e) = v$, then event $e \in E$ has a property $p \in P$ and the value of this property is $v \in \mathcal{U}_V$. If $e \notin dom(\pi_p)$, then event $e$ does not have property $p$ and we write $\pi_p(e) = \perp$ to indicate this.*

The set $E$ refers to the individual events. For example event $e = 35654423$ in Table 1 may be such an event. Note that an event identifier $e \in E$ may be generated implicitly (it has no meaning). $P$ is the set of properties that events may or may not have. For example, $P = \{case, activity, time, resource, cost, type, total\}$ corresponds to the columns in Table 1. $\pi_{case}(35654423) = 1$, $\pi_{activity}(35654423) = A$, and $\pi_{resource}(35654423) = John$ are some of the properties of the first event in Table 1. In the remainder we assume the standard properties *case*, *activity*, and *time* to be defined for all events, i.e., $dom(\pi_{case}) = dom(\pi_{activity}) = dom(\pi_{time}) = E$. For example, we do not allow for events not related to a case. An attribute like *resource* is optional. Note that $\pi$ defines a partial function per property $p$ and missing values are represented as $\pi_p(e) = \perp$. For example, if $\pi_{resource}(e) = \perp$, then $e \notin dom(\pi_{resource})$ implying that $e$ does not have an associated resource.

Assume that $\mathcal{U}_A$ is the set of activities appearing in $EB = (E, P, \pi)$. Given a set of events $E' \subseteq E$, we can compute a multiset of traces $L \in (\mathcal{U}_A)^* \rightarrow \mathbb{N}$ where each trace $\sigma \in L$ corresponds to a case. For example, case 1 in Table 1 can be presented as $\langle A, C, D, E, G\rangle$ and case 2 as $\langle B, C, F, G\rangle$. Most control-flow discovery techniques [1, 11, 12, 16, 30, 18, 24, 25, 28, 31, 41, 54, 60, 61] use such a simple representation as input. This representation ignores concrete timestamps (only the order matters) and abstracts from properties such as *resource*, *cost*, *type*, and *total*.

Note that given an event base, one can derive additional properties. For example, we can take different event properties together, e.g., $\pi_{ar}(e) = (\pi_{activity}(e), \pi_{resource}(e))$. Such derived properties may also be based on other events. For example, $\pi_{st}(e) = min\{\pi_{time}(e') \mid e' \in E \ \wedge \ \pi_{case}(e) = \pi_{case}(e')\}$ is the start time

of the case $e$ belongs to, and $\pi_{sum}(e) = sum\{\pi_{costs}(e') \mid e' \in E \wedge \pi_{case}(e) = \pi_{case}(e')\}$ are the total costs of the case $e$ belongs to. Many useful event attributes can be derived from information inside or outside the initial event base [9]. For example, one can estimate the "stress level" of a resource working on event $e$ by computing the number of queueing activities. In the remainder we assume an event base $EB = (E, P, \pi)$ that includes all properties that may serve as dimensions of the process cube (including derived ones).

## 4 Process Cube Structure

Independent of the event base $EB$ we define the *structure* of the process cube. The structure is fully characterized by the *dimensions* of the cube.

**Definition 3 (Process Cube Structure).** *A* process cube structure *is a triplet* $PCS = (D, type, hier)$ *where:*

- $D$ *is a set of dimensions,*
- $type \in D \rightarrow \mathcal{U}_S$ *is a function defining the possible set of values for each dimension, e.g.,* $type(age) = \{0, 1, 2, \ldots, 120\}$ *for* $age \in D$*, and*
- $hier \in D \rightarrow \mathcal{U}_H$ *defines a hierarchy for each dimension such that for any* $d \in D$*:* $type(d) = \bigcup hier(d)$*. Note that a hierarchy is merely a collection of sets of values.*

A dimension $d \in D$ has a type $type(d)$ and a hierarchy $hier(d)$. $type(d)$ is the set of possible values and typically only a fraction of these values are present in a concrete instance of the process cube. For example, $type(cost) = \mathbb{N}$ allows for infinitely many possible values.

A hierarchy $hier(d)$ is a set of sets. For example $hier(time)$ contains sets such as $T_{2011}$, $T_{2012}$, and $T_{2013}$ each representing all possible timestamps in a particular year.[2] These sets do not need to be disjoint. For example, $hier(time)$ may also contain sets such as $T_{Dec-2012}$ (all possible timestamps in December 2012), $T_{Tue-2012}$ (all Tuesdays in 2012), and $T_{30-12-2012}$ (December 30th 2012). These sets may form a hierarchy based on set inclusion, for example $T_{2012}$ dominates $T_{Dec-2012}$ because $T_{Dec-2012} \subseteq T_{2012}$. Sets may also be partially overlapping, e.g., $T_{Tue-2012} \cap T_{Dec-2012} \neq \emptyset$.

In order to relate an event base and a process cube structure both need to be *compatible*, i.e., dimensions should correspond to properties and concrete event property values need to be of the right type.

**Definition 4 (Compatible).** *A process cube structure* $PCS = (D, type, hier)$ *and an event base* $EB = (E, P, \pi)$ *are compatible if*

- $D \subseteq P$*, i.e., dimensions correspond to properties, and*
- *for any* $d \in D$ *and* $e \in E$*:* $\pi_d(e) \in type(d)$*.*

There are different ways of dealing with *missing values*. The above definition allows for missing values if $\perp \in type(d)$. If $\perp \notin type(d)$, then compatibility implies $dom(\pi_d) = E$.

[2] Note that the notation $T_X$ always refers to a set of timestamps meeting constraint $X$, e.g., $T_{30-12-2012}$ are all timestamps on the specified day.

## 5 Process Cube View

While applying typical OLAP operations such as slice, dice, roll-up and drill-down the event base $EB = (E, P, \pi)$ and process cube structure $PCS = (D, type, hier)$ do not change. It is merely a change of the way event data is viewed. A *process cube view* defines which dimensions are visible and which events are selected.

**Definition 5 (Process Cube View).** *Let $PCS = (D, type, hier)$ be a process cube structure. A* process cube view *is a pair $PCV = (D_{sel}, sel)$ such that:*

- *$D_{sel} \subseteq D$ are the selected dimensions,*
- *$sel \in D \rightarrow \mathcal{U}_H$ is a function selecting the part of the hierarchy considered per dimension. Function sel is such that for any $d \in D$:*
  - *$sel(d) \subseteq hier(d)$, and*
  - *for any $V_1, V_2 \in sel(d)$: $V_1 \subseteq V_2$ implies $V_1 = V_2$.*

A process cube view defines a cube with $k = |D_{sel}|$ dimensions. The maximal number of dimensions is set by $D$, i.e., all dimensions defined in the process cube structure ($D_{sel} \subseteq D$). Function *sel* selects sets of values per dimension (including dimensions not selected in $D_{sel}$). For example, when slicing a cube one decision is removed, but the removed dimension is still used for filtering. Given a dimension $d \in D$, $sel(d)$ defines the elements on the $d$ axis. For example, $sel(time) = \{T_{2011}, T_{2012}, T_{2013}\}$ states that the *time* dimension has three elements. This implies that events before 2011 are filtered out. Moreover, we do not distinguish events based on the month, day or time; only the year matters. $sel(time) = \{T_{2011}, T_{Jan-2012}, T_{Feb-2012}, \ldots, T_{Dec-2012}, T_{2013}\}$ is an alternative view for the *time* dimension. Now the different months of 2012 are distinguished. $sel(d) \subseteq hier(d)$ ensures that the elements of the $d$ dimension are consistent with the process cube structure. The last requirement ($V_1 \subseteq V_2$ implies $V_1 = V_2$) implies that the elements of $sel(d)$ are non-dominating. For example, it would not make sense to have $sel(time) = \{T_{2012}, T_{Jan-2012}\}$ because $T_{Jan-2012} \subseteq T_{2012}$.

As shown in Figure 6, the process cube view can be used to create a *sublog* per cell in the process cube view based on the event base. These sublogs can be viewed as conventional event logs and any process mining technique can be applied to them.

**Definition 6 (Materialized Process Cube View).** *Let process cube structure $PCS = (D, type, hier)$ and event base $EB = (E, P, \pi)$ be compatible. The materialized process cube for some view $PCV = (D_{sel}, sel)$ of PCS is $M_{EB,PCV} = \{(c, events(c)) \mid c \in cells\}$ with $cells = \{c \in D_{sel} \rightarrow \mathcal{U}_S \mid \forall_{d \in D_{sel}}\ c(d) \in sel(d)\}$ being the cells of the cube and $events(c) = \{e \in E \mid \forall_{d \in D_{sel}}\ \pi_d(e) \in c(d) \ \wedge\ \forall_{d \in D}\ \pi_d(e) \in \bigcup sel(d)\}$ the set of events per cell.*

*cells* is the collection of cells of the cube. A $c \in cells$ is an assignment of each visible dimension to precisely one element of that dimension, e.g., $c(time) = T_{Jan-2012}$, $c(resource) = \{John, Pete\}$, and $c(type) = \{gold\}$. $events(c)$ are all

events corresponding to cell $c$ (first requirement: $\forall_{d \in D_{sel}}\ \pi_d(e) \in c(d)$) and not filtered out (second requirement: $\forall_{d \in D}\ \pi_d(e) \in \bigcup sel(d)$).

Definition 6 provides the interface to existing process discovery [1, 11, 12, 16, 30, 18, 24, 25, 28, 31, 41, 54, 60, 61] and conformance checking [6, 13, 14, 15, 22, 29, 31, 42, 43, 51, 59] techniques. $M_{EB,PCV}$ defines how to compute an event log (called sublog) per cell. As shown in Figure 6, these sublogs can be used to compute results per cell.

Note that the materialized process cube view $M_{EB,PCV}$ may be constructed on-the-fly or pre-computed. Existing OLAP tools often materialize views in order to enable interactive analysis. However, for process mining techniques it is typically not known how to do this efficiently.

## 6 Slice and Dice

Next we consider the classical OLAP operations in the context of our process cubes.

The *slice operation* produces a sliced OLAP cube by allowing the analyst to pick specific value for one of the dimensions. For example, for sales data one can slice the cube for location "Eindhoven", i.e., the location dimension is removed from the cube and only sales of the stores in Eindhoven are considered. Slicing the cube for the year "2012" implies removing the time dimension and only considering sales in 2012. The *dice operation* produces a subcube by allowing the analyst to pick specific values for multiple dimensions. For example, one could dice the sales OLAP cube for years "2012" and "2013" and locations "Eindhoven" and "Amsterdam". No dimensions are removed, but only sales in 2012 and 2013 in stores in Eindhoven and Amsterdam are considered.

Given the earlier formalizations, we can easily define the slice operation for process cubes.

**Definition 7 (Slice).** *Let $PCS = (D, type, hier)$ be a process cube structure and $PCV = (D_{sel}, sel)$ a view of $PCS$. For any $d \in D_{sel}$ and $V \in sel(d)$: $slice_{d,V}(PCV) = (D'_{sel}, sel')$ with $D'_{sel} = D_{sel} \setminus \{d\}$, $sel'(d) = \{V\}$, and $sel'(d') = sel(d')$ for $d' \in D \setminus \{d\}$.*

$slice_{d,V}(PCV)$ produces a new process cube view. Note that $d$ is no longer a visible dimension: $d \notin D'_{sel}$. At the same time $d$ is used to filter events: only events $e$ with $\pi_d(e) \in V$ are considered in the new view.

**Definition 8 (Dice).** *Let $PCS = (D, type, hier)$ be a process cube structure and $PCV = (D_{sel}, sel)$ a view of $PCS$. Let $res \in D_{sel} \not\rightarrow \mathcal{U}_H$ be a restriction such for any $d \in dom(res)$: $res(d) \subseteq sel(d)$. $dice_{res}(PCV) = (D_{sel}, sel')$ with $sel'(d) = res(d)$ for $d \in dom(res)$ and $sel'(d) = sel(d)$ for $d \in D \setminus dom(res)$.*

$dice_{res}(PCV)$ produces a process cube view having the original dimensions. $res \in D_{sel} \not\rightarrow \mathcal{U}_H$ restricts selected dimensions. For example, if $res(time) = \{T_{Jan-2012}, T_{Jan-2013}\}$, $res(resource) = \{\{John\}, \{Pete\}\}$, and $res(type) =$

$\{\{gold, silver\}\}$, then $dice_{res}(PCV)$ restricts the time dimension to two elements (2012 and 2013), the resource dimension to two elements (John and Pete), and the customer type dimension to one element (both gold and silver customers).

## 7 Roll-Up and Drill-Down

Roll-up and drill-down operations do not remove any events but change the level of granularity of a particular dimension. For example, before drilling down $sel(time) = \{T_{2011}, T_{2012}, T_{2013}\}$ and after drilling down $sel'(time) = \{T_{2011}, T_{Jan-2012}, T_{Feb-2012}, \ldots, T_{Dec-2012}, T_{2013}\}$. Rolling up (sometimes referred to as drilling up) is the reverse. For example, $sel(type) = \{\{gold\}, \{silver\}\}$ is rolled up into $sel'(type) = \{\{gold, silver\}\}$.

**Definition 9 (Change Granularity).** *Let $PCS = (D, type, hier)$ be a process cube structure and $PCV = (D_{sel}, sel)$ a view of $PCS$. Let $d \in D_{sel}$ and $H \in \mathcal{U}_H$ such that:*

- *$H \subseteq hier(d)$,*
- *$\bigcup H = \bigcup sel(d)$, and*
- *for any $V_1, V_2 \in H$: $V_1 \subseteq V_2$ implies $V_1 = V_2$.*

*$chgr_{d,H}(PCV) = (D_{sel}, sel')$ with $sel'(d) = H$, and $sel'(d') = sel(d')$ for $d' \in D \setminus \{d\}$.*

$chgr_{d,H}(PCV)$ yields a process cube view with the original dimensions $D_{sel}$. However, dimension $d$ is reorganized in such a way that the result is indeed a view (e.g., elements are not dominating and consistent with the process cube structure) and the set of possible values is unchanged $\bigcup sel'(d) = \bigcup sel(d)$.

## 8 Conclusion

In this paper, we formalized the notion of process cubes. It gives end users the opportunity to analyze and explore processes interactively on the basis of a multidimensional view on event data. There is no need to extract event logs beforehand like in traditional process mining approaches. Although an initial prototype implementation supporting the main ideas in this paper has been realized [39], many challenges remain. In the remainder, we discuss some of these challenges.

### 8.1 Comparing and Visualizing Different Cells

First of all, there is the challenge of comparing and visualizing different cells. How to visualize this in an effective manner? Unlike the numerical values shown in traditional OLAP cubes, we need to visualize models that cannot be reduced to simple numbers. Two models many be similar, but their visualizations may be unrelated. This is not just a matter of lay-out. Two process models that are

similar from a representational point of view may have very different behaviors and two process models that are different from a representational point of view may have very similar behaviors [1]. Here, we can benefit from research on *configurable process models*. A configurable process model represents a *family* of process models, that is, a model that through configuration can be customized for a particular setting [32, 47, 50, 52]. Process models belonging to such a family need to be related, just like cells in a process cube need to be related to allow for comparison.

Given a process cube, we suggest to visualize the different models with respect to a *cornerstone model*. The different cell models are visualized as edit operations on the cornerstone model. Typical edit operations are: add/remove activity, add/remove edge, hide/insert activity, swap activities, and sequentialize/parallelize activities. These edit operations have costs and are minimized to find the shortest path from the cornerstone model to a particular cell model. Moreover, the edit operations for the different cells are aligned to make the overall understanding of the process cube as simple as possible. One may consider a restricted set of edit operations for block-structured process models [57] to simplify comparison.

There are different approaches to obtain the cornerstone model. The cornerstone model may be selected by the user or may be the first or last model in an array of cells. Moreover, the model may be the Greatest Common Divisor (GCD) or the Least Common Multiple (LCM) of the collection of process models considered [7]. The GCD captures the common parts of the cell models, i.e., all cell models are extensions of the GCD. The LCM embeds all cell models, i.e., all models are restrictions of the LCM. These notions are based on the observation that "hiding" and "blocking" are the essential operators needed for defining inheritance with respect to behavior [8]. The cornerstone model may also be the model closest to all cell models (minimal average edit distance) [38].

### 8.2 Computing Sublogs and Models Per Cell

Second, there is the problem of performance. The OLAP operations need to be instantaneous to allow for direct user interaction. To realize this, cell results may be pre-computed (materialization of event data and process mining results, e.g., models). However, this may be infeasible in case of many sparse dimensions. Hence, it may be better to do this on-the-fly.

Figure 5 already illustrated the notion of splitting/merging cells horizontally/vertically. We want to do this efficiently for *both* logs and models.

When merging cells one can discover the process model from scratch using the merged event log. As this can be time-consuming, it may be better to merge the process models. Various approaches for merging process models have been proposed in literature [33, 48]. However, these approaches only merge vertically (cf. Figure 5), whereas we also need to support the horizontal merge. Moreover, existing approaches for model merging are not taking into account the event log. Therefore, we would like to develop hybrid approaches that exploit both the

existing models and the log to create a merged model that is as close as possible to the original models and the merged event log.

When splitting cells one can discover a process model for each of the smaller event logs. Again this may be time-consuming. Moreover, after splitting, the resulting event logs may be too small to create reliable models. Therefore, we would like to develop hybrid approaches that exploit both the original model and the smaller event logs to create a model for each new cell. For example, the original model may be projected using information from the sublog.

When splitting and merging process cells, one may need to preserve existing relationships between model and event log, e.g., so-called *alignments* [6, 13] need to be split and merged without losing any connections.

### 8.3 Concept Drift

The time dimension of a process cube has specific properties that can be exploited. For example, the hierarchy of the time dimension can be shared among different applications. Moreover, time introduces particular challenges. For example, processes often change while being analyzed. Therefore, *concept drift* is mentioned as one of the main challenges in the Process Mining Manifesto [36]. Concept drift was been investigated in the context of various data mining problems [62, 37]. In [19] the problem was first investigated in the context of process mining. However, many challenges remain [19, 26], e.g., dealing with incremental drifts and mixtures of periodic drifts.

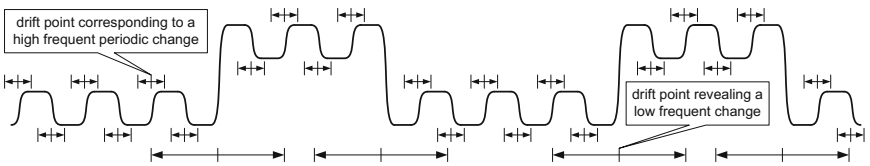



**Fig. 7.** A periodically changing processes with two types of drift at different time scales

Note that the time window dimension in Figure 4 is different from the case type and event class dimensions. In case of short-running cases, we can associate whole cases to time windows. In case of long-running cases, we need to associate individual events to time windows as the process may change while the instance is running. Using carefully selected feature vectors we can analyze drifts using sliding time windows: statistical hypothesis testing will reveal drifts if there are significant differences between two successive windows. A complication is that different types of drifts may be intertwined as illustrated by Figure 7. The drift points are depicted by the bars splitting the double headed arrows: the split arrows represent two consecutive time windows having significant differences. We would also like to relate process changes to contextual elements captured by the cube's dimensions. For example, the time of the day, the weather, the

workload, or the type of customer may influence the way cases are handled. As an additional complication, classical conformance notions such as fitness, generalization, and precision [1, 6] cannot be applied to processes that change as one needs to judge the result with respect to a particular time window. Concept drift is also related to *on-the-fly process discovery* [21] where event streams are not stored.

### 8.4 Distributed Process Mining

Today, there are many types of distributed systems, i.e., systems composed of multiple autonomous computational entities communicating through a network. The terms grid computing, multicore CPU systems, manycore GPU systems, cluster computing, and cloud computing all refer to technologies where different resources are used concurrently to improve performance and scalability. Most data mining techniques can be distributed [23], e.g., there are various techniques for distributed classification, distributed clustering, and distributed association rule mining [17]. These techniques cannot be applied to process mining because events belong to cases and the ordering of events matters. Yet, there is an obvious need for distributed process mining using more efficient and effective discovery techniques. Process mining tasks become challenging when there are hundreds or even thousands of different activities and millions of cases. Typically, process mining algorithms are linear in the number of cases and exponential in the number of different activities.

Process cubes partition event data and therefore may enable *divide-and-conquer approaches* that decompose the event log based on splitting/merging cells horizontally/vertically [3]. Thus was already illustrated using Figure 5. We are particularly interested in splitting logs horizontally. Thus far we have developed horizontal divide-and-conquer approaches based on SESEs [45, 44], passages [2, 58], and maximal decompositions [5] as a decomposition strategy. As demonstrated in [4, 5] these are merely examples of the broad spectrum of possible techniques to decompose process mining problems. Given the incredible growth of event data, there is an urgent need to explore and investigate the entire spectrum in more detail. Hopefully, such techniques can be used to speed-up OLAP-like operations on process cubes.

**Acknowledgements.** This work was supported by the Basic Research Program of the National Research University Higher School of Economics (HSE). The author would also like to thank Tatiana Mamaliga for her work on realizing ProCube, a prototype process cube implementation based on ProM and Palo (supervised by the author and Joos Buijs).

## References

1. van der Aalst, W.M.P.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer, Berlin (2011)

2. van der Aalst, W.M.P.: Decomposing Process Mining Problems Using Passages. In: Haddad, S., Pomello, L. (eds.) PETRI NETS 2012. LNCS, vol. 7347, pp. 72–91. Springer, Heidelberg (2012)
3. van der Aalst, W.M.P.: Distributed Process Discovery and Conformance Checking. In: de Lara, J., Zisman, A. (eds.) FASE 2012. LNCS, vol. 7212, pp. 1–25. Springer, Heidelberg (2012)
4. van der Aalst, W.M.P.: A General Divide and Conquer Approach for Process Mining. In: Ganzha, M., Maciaszek, L., Paprzycki, M. (eds.) Federated Conference on Computer Science and Information Systems (FedCSIS 2013), pp. 1–10. IEEE Computer Society (2013)
5. van der Aalst, W.M.P.: Decomposing Petri Nets for Process Mining: A Generic Approach. Distributed and Parallel Databases 31(4), 471–507 (2013)
6. van der Aalst, W.M.P., Adriansyah, A., van Dongen, B.: Replaying History on Process Models for Conformance Checking and Performance Analysis. WIREs Data Mining and Knowledge Discovery 2(2), 182–192 (2012)
7. van der Aalst, W.M.P., Basten, T.: Identifying Commonalities and Differences in Object Life Cycles using Behavioral Inheritance. In: Colom, J.-M., Koutny, M. (eds.) ICATPN 2001. LNCS, vol. 2075, pp. 32–52. Springer, Heidelberg (2001)
8. van der Aalst, W.M.P., Basten, T.: Inheritance of Workflows: An Approach to Tackling Problems Related to Change. Theoretical Computer Science 270(1-2), 125–203 (2002)
9. van der Aalst, W.M.P., Dustdar, S.: Process Mining Put into Context. IEEE Internet Computing 16(1), 82–86 (2012)
10. van der Aalst, W.M.P., Reijers, H.A., Song, M.: Discovering Social Networks from Event Logs. Computer Supported Cooperative Work 14(6), 549–593 (2005)
11. van der Aalst, W.M.P., Rubin, V., Verbeek, H.M.W., van Dongen, B.F., Kindler, E., Günther, C.W.: Process Mining: A Two-Step Approach to Balance Between Underfitting and Overfitting. Software and Systems Modeling 9(1), 87–111 (2010)
12. van der Aalst, W.M.P., Weijters, A.J.M.M., Maruster, L.: Workflow Mining: Discovering Process Models from Event Logs. IEEE Transactions on Knowledge and Data Engineering 16(9), 1128–1142 (2004)
13. Adriansyah, A., van Dongen, B., van der Aalst, W.M.P.: Conformance Checking using Cost-Based Fitness Analysis. In: Chi, C.H., Johnson, P. (eds.) IEEE International Enterprise Computing Conference (EDOC 2011), pp. 55–64. IEEE Computer Society (2011)
14. Adriansyah, A., van Dongen, B.F., van der Aalst, W.M.P.: Towards Robust Conformance Checking. In: zur Muehlen, M., Su, J. (eds.) BPM 2010 Workshops. LNBIP, vol. 66, pp. 122–133. Springer, Heidelberg (2011)
15. Adriansyah, A., Sidorova, N., van Dongen, B.F.: Cost-based Fitness in Conformance Checking. In: International Conference on Application of Concurrency to System Design (ACSD 2011), pp. 57–66. IEEE Computer Society (2011)
16. Agrawal, R., Gunopulos, D., Leymann, F.: Mining Process Models from Workflow Logs. In: Schek, H.-J., Saltor, F., Ramos, I., Alonso, G. (eds.) EDBT 1998. LNCS, vol. 1377, pp. 469–483. Springer, Heidelberg (1998)
17. Agrawal, R., Shafer, J.C.: Parallel Mining of Association Rules. IEEE Transactions on Knowledge and Data Engineering 8(6), 962–969 (1996)
18. Bergenthum, R., Desel, J., Lorenz, R., Mauser, S.: Process Mining Based on Regions of Languages. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 375–383. Springer, Heidelberg (2007)

19. Bose, R.P.J.C., van der Aalst, W.M.P., Žliobaitė, I., Pechenizkiy, M.: Handling Concept Drift in Process Mining. In: Mouratidis, H., Rolland, C. (eds.) CAiSE 2011. LNCS, vol. 6741, pp. 391–405. Springer, Heidelberg (2011)
20. Buijs, J.C.A.M., van Dongen, B.F., van der Aalst, W.M.P.: Towards cross-organizational process mining in collections of process models and their executions. In: Daniel, F., Barkaoui, K., Dustdar, S. (eds.) BPM Workshops 2011, Part II. LNBIP, vol. 100, pp. 2–13. Springer, Heidelberg (2012)
21. Burattin, A., Sperduti, A., van der Aalst, W.M.P.: Heuristics Miners for Streaming Event Data. CoRR, abs/1212.6383 (2012)
22. Calders, T., Guenther, C., Pechenizkiy, M., Rozinat, A.: Using Minimum Description Length for Process Mining. In: ACM Symposium on Applied Computing (SAC 2009), pp. 1451–1455. ACM Press (2009)
23. Cannataro, M., Congiusta, A., Pugliese, A., Talia, D., Trunfio, P.: Distributed Data Mining on Grids: Services, Tools, and Applications. IEEE Transactions on Systems, Man, and Cybernetics, Part B 34(6), 2451–2465 (2004)
24. Carmona, J., Cortadella, J.: Process Mining Meets Abstract Interpretation. In: Balcázar, J.L., Bonchi, F., Gionis, A., Sebag, M. (eds.) ECML PKDD 2010, Part I. LNCS, vol. 6321, pp. 184–199. Springer, Heidelberg (2010)
25. Carmona, J., Cortadella, J., Kishinevsky, M.: A Region-Based Algorithm for Discovering Petri Nets from Event Logs. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240, pp. 358–373. Springer, Heidelberg (2008)
26. Carmona, J., Gavaldà, R.: Online techniques for dealing with concept drift in process mining. In: Hollmén, J., Klawonn, F., Tucker, A. (eds.) IDA 2012. LNCS, vol. 7619, pp. 90–102. Springer, Heidelberg (2012)
27. Chaudhuri, S., Dayal, U.: An Overview of Data Warehousing and OLAP Technology. ACM Sigmod Record 26(1), 65–74 (1997)
28. Cook, J.E., Wolf, A.L.: Discovering Models of Software Processes from Event-Based Data. ACM Transactions on Software Engineering and Methodology 7(3), 215–249 (1998)
29. Cook, J.E., Wolf, A.L.: Software Process Validation: Quantitatively Measuring the Correspondence of a Process to a Model. ACM Transactions on Software Engineering and Methodology 8(2), 147–176 (1999)
30. Gaaloul, W., Gaaloul, K., Bhiri, S., Haller, A., Hauswirth, M.: Log-Based Transactional Workflow Mining. Distributed and Parallel Databases 25(3), 193–240 (2009)
31. Goedertier, S., Martens, D., Vanthienen, J., Baesens, B.: Robust Process Discovery with Artificial Negative Events. Journal of Machine Learning Research 10, 1305–1340 (2009)
32. Gottschalk, F., van der Aalst, W.M.P., Jansen-Vullers, M.H., La Rosa, M.: Configurable Workflow Models. International Journal of Cooperative Information Systems 17(2), 177–221 (2008)
33. Gottschalk, F., Wagemakers, T.A.C., Jansen-Vullers, M.H., van der Aalst, W.M.P., La Rosa, M.: Configurable Process Models: Experiences From a Municipality Case Study. In: van Eck, P., Gordijn, J., Wieringa, R. (eds.) CAiSE 2009. LNCS, vol. 5565, pp. 486–500. Springer, Heidelberg (2009)
34. Günther, C.W., van der Aalst, W.M.P.: Fuzzy Mining: Adaptive Process Simplification Based on Multi-perspective Metrics. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 328–343. Springer, Heidelberg (2007)
35. Hilbert, M., Lopez, P.: The World's Technological Capacity to Store, Communicate, and Compute Information. Science 332(6025), 60–65 (2011)

36. IEEE Task Force on Process Mining. Process Mining Manifesto. In: Daniel, F., Barkaoui, K., Dustdar, S. (eds.) Business Process Management Workshops. LNBIP, vol. 99, pp. 169–194. Springer, Berlin (2012)
37. van Leeuwen, M., Siebes, A.: StreamKrimp: Detecting Change in Data Streams. In: Daelemans, W., Goethals, B., Morik, K. (eds.) ECML PKDD 2008, Part I. LNCS (LNAI), vol. 5211, pp. 672–687. Springer, Heidelberg (2008)
38. Li, C., Reichert, M., Wombacher, A.: The MINADEPT Clustering Approach for Discovering Reference Process Models Out of Process Variants. International Journal of Cooperative Information Systems 19(3-4), 159–203 (2010)
39. Mamaliga, T.: Realizing a Process Cube Allowing for the Comparison of Event Data. Master's thesis, Eindhoven University of Technology, Eindhoven (2013)
40. Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., Byers, A.: Big Data: The Next Frontier for Innovation, Competition, and Productivity. McKinsey Global Institute (2011)
41. Alves de Medeiros, A.K., Weijters, A.J.M.M., van der Aalst, W.M.P.: Genetic Process Mining: An Experimental Evaluation. Data Mining and Knowledge Discovery 14(2), 245–304 (2007)
42. Muñoz-Gama, J., Carmona, J.: A Fresh Look at Precision in Process Conformance. In: Hull, R., Mendling, J., Tai, S. (eds.) BPM 2010. LNCS, vol. 6336, pp. 211–226. Springer, Heidelberg (2010)
43. Munoz-Gama, J., Carmona, J.: Enhancing Precision in Process Conformance: Stability, Confidence and Severity. In: Chawla, N., King, I., Sperduti, A. (eds.) IEEE Symposium on Computational Intelligence and Data Mining (CIDM 2011), Paris, France, pp. 184–191. IEEE (April 2011)
44. Munoz-Gama, J., Carmona, J., van der Aalst, W.M.P.: Conformance Checking in the Large: Partitioning and Topology. In: Daniel, F., Wang, J., Weber, B. (eds.) BPM 2013. LNCS, vol. 8094, pp. 130–145. Springer, Heidelberg (2013)
45. Munoz-Gama, J., Carmona, J., van der Aalst, W.M.P.: Hierarchical Conformance Checking of Process Models Based on Event Logs. In: Colom, J.-M., Desel, J. (eds.) PETRI NETS 2013. LNCS, vol. 7927, pp. 291–310. Springer, Heidelberg (2013)
46. Ribeiro, J.T.S., Weijters, A.J.M.M.: Event Cube: Another Perspective on Business Processes. In: Meersman, R., Dillon, T., Herrero, P., Kumar, A., Reichert, M., Qing, L., Ooi, B.-C., Damiani, E., Schmidt, D.C., White, J., Hauswirth, M., Hitzler, P., Mohania, M. (eds.) OTM 2011, Part I. LNCS, vol. 7044, pp. 274–283. Springer, Heidelberg (2011)
47. La Rosa, M., Dumas, M., ter Hofstede, A., Mendling, J.: Configurable Multi-Perspective Business Process Models. Information Systems 36(2), 313–340 (2011)
48. La Rosa, M., Dumas, M., Uba, R., Dijkman, R.M.: Business Process Model Merging: An Approach to Business Process Consolidation. ACM Transactions on Software Engineering and Methodology 22(2) (2012)
49. La Rosa, M., Reijers, H.A., van der Aalst, W.M.P., Dijkman, R.M., Mendling, J., Dumas, M., Garcia-Banuelos, L.: APROMORE: An Advanced Process Model Repository. Expert Systems With Applications 38(6), 7029–7040 (2011)
50. Rosemann, M., van der Aalst, W.M.P.: A Configurable Reference Modelling Language. Information Systems 32(1), 1–23 (2007)
51. Rozinat, A., van der Aalst, W.M.P.: Conformance Checking of Processes Based on Monitoring Real Behavior. Information Systems 33(1), 64–95 (2008)
52. Schnieders, A., Puhlmann, F.: Variability Mechanisms in E-Business Process Families. In: Abramowicz, W., Mayr, H.C. (eds.) Proceedings of the 9th International Conference on Business Information Systems (BIS 2006). LNI, vol. 85, pp. 583–601. GI (2006)

53. Sheth, A.: A New Landscape for Distributed and Parallel Data Management. Distributed and Parallel Databases 30(2), 101–103 (2012)
54. Solé, M., Carmona, J.: Process Mining from a Basis of Regions. In: Lilius, J., Penczek, W. (eds.) PETRI NETS 2010. LNCS, vol. 6128, pp. 226–245. Springer, Heidelberg (2010)
55. Song, M., van der Aalst, W.M.P.: Supporting Process Mining by Showing Events at a Glance. In: Chari, K., Kumar, A. (eds.) Proceedings of 17th Annual Workshop on Information Technologies and Systems (WITS 2007), Montreal, Canada, pp. 139–145 (December 2007)
56. Song, M., van der Aalst, W.M.P.: Towards Comprehensive Support for Organizational Mining. Decision Support Systems 46(1), 300–317 (2008)
57. Vanhatalo, J., Völzer, H., Koehler, J.: The Refined Process Structure Tree. Data and Knowledge Engineering 68(9), 793–818 (2009)
58. Verbeek, H.M.W., van der Aalst, W.M.P.: Decomposing Replay Problems: A Case Study. BPM Center Report BPM-13-09, BPMcenter.org (2013)
59. Weerdt, J.D., Backer, M.D., Vanthienen, J., Baesens, B.: A Robust F-measure for Evaluating Discovered Process Models. In: Chawla, N., King, I., Sperduti, A. (eds.) IEEE Symposium on Computational Intelligence and Data Mining (CIDM 2011), Paris, France, pp. 148–155. IEEE (April 2011)
60. Weijters, A.J.M.M., van der Aalst, W.M.P.: Rediscovering Workflow Models from Event-Based Data using Little Thumb. Integrated Computer-Aided Engineering 10(2), 151–162 (2003)
61. van der Werf, J.M.E.M., van Dongen, B.F., Hurkens, C.A.J., Serebrenik, A.: Process Discovery using Integer Linear Programming. Fundamenta Informaticae 94, 387–412 (2010)
62. Widmer, G., Kubat, M.: Learning in the Presence of Concept Drift and Hidden Contexts. Machine Learning 23, 69–101 (1996)

# APQL: A Process-Model Query Language

Arthur H.M. ter Hofstede[1,2], Chun Ouyang[1], Marcello La Rosa[1,3], Liang Song[4], Jianmin Wang[4], and Artem Polyvyanyy[1]

[1] Queensland University of Technology, Brisbane, Australia
{a.terhofstede,c.ouyang,m.larosa,artem.polyvyanyy}@qut.edu.au
[2] Eindhoven University of Technology, Eindhoven, The Netherlands
[3] NICTA Queensland Lab, Brisbane, Australia
[4] School of Software, Tsinghua University, Beijing, China
songliang08@mails.thu.edu.cn, jimwang@tsinghua.edu.cn

**Abstract.** As business process management technology matures, organisations acquire more and more business process models. The management of the resulting collections of process models poses real challenges. One of these challenges concerns model retrieval where support should be provided for the formulation and efficient execution of business process model queries. As queries based on only structural information cannot deal with all querying requirements in practice, there should be support for queries that require knowledge of process model semantics. In this paper we formally define a process model query language that is based on semantic relationships between tasks in process models and is independent of any particular process modelling notation.

**Keywords:** business process model, process model collection, business process model query, query language.

## 1 Introduction

With the increasing maturity of business process management, more and more organisations need to manage large numbers of business process models, and among these may be models of high complexity. Processes may be defined along the entire value chain and over time a business may gather hundreds and even thousands of business process models. As an example consider Suncorp, one of the largest Australian insurers. Over the years, Suncorp have gone through a number of organizational mergers and acquisitions, as a result of which the company has accumulated over 3,000 process models for the various lines of insurance. In this context, support for business process retrieval, e.g. for the purposes of process reuse or process standardization, is a challenging proposition.

Several query languages exist that can be used to retrieve process models from a repository, e.g. BPMN-Q [1] or BP-QL [2,3]. These languages are based on syntactic relationships between tasks and not on their semantic relationships. However, to deal with all querying requirements in practice, it is not enough to rely on only structural information of the process models but often requires knowledge of process model semantics. Consider for example the two process

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-319-02922-1_10

M. Song, M.T. Wynn, and J. Liu (Eds.): AP-BPM 2013, LNBIP 159, pp. 23–38, 2013.

models in Fig. 1. They describe two variants of a business process for opening bank accounts using the BPMN notation [4]. These two variants could capture the way an account is opened in two different states where a bank operates, and could be part of a collection of various process models in all states where the bank operates. Assume that a business analyst needs to find out which branches always require an assessment of the customer's credit history when opening an account. In this case, only using the structural relationships between tasks, we cannot discern between the two variants, i.e. we would retrieve them both, since in both models there is a path from task "Receive customer request" to task "Analyse customer credit history". However, based on semantic relationships between tasks, we can observe that task "Analyse customer credit history" follows task "Receive customer request" in all instances of the first process variant, but this is not the case for one instance of the second variant (the one with task "Open VIP account"). Thus we can correctly exclude the second process variant from the results of the query, and return the first variant only.

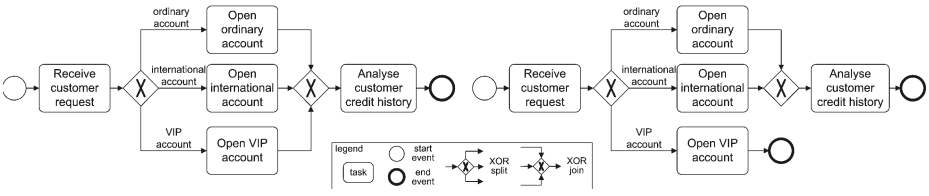



**Fig. 1.** Two variants of a business process for opening bank accounts

In light of the above, we aim to address the development of a business process model query language based on semantic relationships between tasks in process models. We do so by proposing a new query language, namely *A Process-model Query Language (APQL)*, for retrieving required process models from model collections, e.g. process model repositories. This language relies on a number of basic temporal relationships between tasks which can be composed to obtain complex relationships between them. These predicates allow us to express queries that can discriminate over single process instances or task instances.

In this paper, we define both the syntax (Sect. 2) and semantics (Sect. 3) of APQL, provide examples to assist in understanding of the language definition (Sect. 4), discuss related work (Sect. 5), and finally conclude the paper (Sect. 6).

## 2 The Syntax of APQL

APQL is designed as a process model query language that is independent of the actual process modelling language used. This is important as in practice a variety of modelling languages are used (e.g. BPMN, EPCs) and the language should be generally applicable. Another important fact is that process models have a semantics and it should be possible to exploit this semantics when querying.

Based on the above design rationale, we define a set of 20 basic predicates to capture, in business process models, the occurrences of tasks as well as the semantic relationships between tasks. Below, the first two predicates capture the occurrence of a task $t$ in *some* or *every* execution of a given process model $r$.

1. $posoccur(t, r)$: some execution of $r$ exists where at least one instance of $t$ occurs.
2. $alwoccur(t, r)$: in every execution of $r$, at least one instance of $t$ occurs.

The next two predicates capture the *exclusive* and *concurrent* relationships between task occurrences. Note that these two predicates do not assume that an instance of $t_1$ or $t_2$ should eventually occur in a given process model $r$.

3. $exclusive(t_1, t_2, r)$: in every execution of $r$, it is never possible that an instance of $t_1$ and an instance of $t_2$ both occur.
4. $concur(t_1, t_2, r)$: $t_1$ and $t_2$ are not causally related, and in every execution of $r$, if an instance of $t_1$ occurs then an instance of $t_2$ occurs and vice versa.

Then we consider various forms of causal relationship between task occurrences. The relationship can be *precedence* (*pred*) or *succession* (*succ*), where one task may occur *immediately* or *eventually* preceding or succeeding another task. It may hold for *any* or *every* occurrence of the tasks in *some* or *every* process execution. Combining all these considerations results in 16 forms of causal relationships which are captured by the remaining 16 basic predicates as follows.

Let $\Phi$ be one of the following intermediate predicates,

1. $succ_{any}(t_1, t_2, i)$: in process execution $i$, at least one instance of $t_1$ occurs and is eventually succeeded by an instance of $t_2$ (e.g. $...t_1...t_2...$).
2. $succ_{every}(t_1, t_2, i)$: in process execution $i$, at least one instance of $t_1$ occurs and every instance of $t_1$ is eventually succeeded by an instance of $t_2$ (e.g. $t_1...t_1...t_2$).
3. $pred_{any}(t_1, t_2, i)$: in process execution $i$, at least one instance of $t_1$ occurs and is eventually preceded by an instance of $t_2$ (e.g. $...t_2...t_1...$).
4. $pred_{every}(t_1, t_2, i)$: in process execution $i$, at least one instance of $t_1$ occurs and every instance of $t_1$ is eventually preceded by an instance of $t_2$ (e.g. $t_2...t_1...t_1$).
5. $isucc_{any}(t_1, t_2, i)$: in process execution $i$, at least one instance of $t_1$ occurs and is immediately succeeded by an instance of $t_2$ (e.g. $...t_1t_2...$).
6. $isucc_{every}(t_1, t_2, i)$: in process execution $i$, at least one instance of $t_1$ occurs and every instance of $t_1$ is immediately succeeded by an instance of $t_2$ (e.g. $t_1t_2...t_1t_2$).
7. $ipred_{any}(t_1, t_2, i)$: in process execution $i$, at least one instance of $t_1$ occurs and is immediately preceded by an instance of $t_2$ (e.g. $...t_2t_1...$).
8. $ipred_{every}(t_1, t_2, i)$: in process execution $i$, at least one instance of $t_1$ occurs and every instance of $t_1$ is immediately preceded by an instance of $t_2$ (e.g. $t_2t_1...t_2t_1$).

Then

- $\Phi^{\forall}(t_1, t_2, r)$: $\Phi(t_1, t_2, i)$ holds for every process execution $i$ of process model $r$, i.e. $\Phi(t_1, t_2, i)$ *always* holds in process $r$, and
- $\Phi^{\exists}(t_1, t_2, r)$: there *exists* some process execution $i$ of process model $r$ where $\Phi(t_1, t_2, i)$ holds, i.e. it is *possible* that $\Phi(t_1, t_2, i)$ holds in process $r$.

Next, the syntax of APQL is defined in the form of an abstract syntax, the advantages of which over a concrete syntax have been espoused by Meyer [5].

In essence, in an abstract syntax we can avoid committing ourselves prematurely to specific choices for keywords or to the order of various statements.

In APQL a query is a sequence of *Assignments* combined with a *Predicate*.

$$\begin{aligned} \mathit{Query} &\triangleq s : \mathit{Assignments}; p : \mathit{Predicate} \\ \mathit{Assignments} &\triangleq \mathit{Assignment}^* \end{aligned}$$

The result is those process models that satisfy the *Predicate*. An *Assignment* assigns a *TaskSet* to a variable and when evaluating the *Predicate* every variable is replaced by the corresponding *TaskSet* (via the identifier of such task set).

$$\begin{aligned} \mathit{Assignment} &\triangleq v : \mathit{Varname}; ts : \mathit{TaskSet} \\ \mathit{Varname} &\triangleq \mathit{identifier} \end{aligned}$$

A *TaskSet* can be an enumeration of tasks or defined over other *TaskSet*s by *Construction* or *Application*. It can also be defined through a variable, a *TaskSetVar*.

$$\mathit{TaskSet} \triangleq \mathit{SetofTasks} \mid \mathit{Construction} \mid \mathit{Application} \mid \mathit{TaskSetVar}$$

A *Task* can be defined as a combination of a *TaskLabel* (a string) and a *SimDegree* (a real number). The idea is that one may be interested in *Tasks* of which the task label bears a certain degree of similarity to a given activity name. There are a number of definitions in the literature concerning label similarity and for a concrete implementation of the language one has to commit to one of these.

$$\begin{aligned} \mathit{SetofTasks} &\triangleq \mathit{Task}^* \\ \mathit{Task} &\triangleq l : \mathit{TaskLabel}; d : \mathit{SimDegree} \\ \mathit{TaskLabel} &\triangleq \mathit{string} \\ \mathit{SimDegree} &\triangleq \mathit{real}[0..1] \end{aligned}$$

A *TaskSetVar* is simply a variable that carries the identifier of the set of the tasks. Such a task set may be used in assignments.

$$\mathit{TaskSetVar} \triangleq \mathit{identifier}$$

A *TaskSet* can be composed from other *TaskSet*s through the application of the well-known set operators such as *union*, *difference*, and *intersection*. Another way to construct a *TaskSet* is by the application of a *TaskCompOp* (i.e. one of the basic predicates introduced earlier, but now interpreted as a function) on another *TaskSet*. In that case we have to specify whether we are interested in the tasks that have that particular relation with *all* or with *any* of the tasks in the *TaskSet*. For example, an application with *TaskSet* $S$, *TaskCompOp* *PosSuccAny* (i.e. $succ_{any}{}^{\exists}$) and *AnyAll* (indicator) *All*, is to yield those tasks that any instance of such a task succeeds an instance of each task in $S$ in some process execution.

$$\begin{aligned} \mathit{Construction} &\triangleq ts_1, ts_2 : \mathit{TaskSet}; o : \mathit{Set\_Op} \\ \mathit{Set\_Op} &\triangleq \mathit{Union} \mid \mathit{Difference} \mid \mathit{Intersection} \\ \mathit{Application} &\triangleq ts : \mathit{TaskSet}; o : \mathit{TaskCompOp}; a : \mathit{AnyAll} \end{aligned}$$

$$
\begin{aligned}
\mathit{TaskCompOp} &\triangleq \mathit{Exclusive} \mid \mathit{Concur} \mid \\
&\quad \mathit{AlwSuccAny} \mid \mathit{AlwSuccEvery} \mid \mathit{AlwPredAny} \mid \mathit{AlwPredEvery} \mid \\
&\quad \mathit{PosSuccAny} \mid \mathit{PosSuccEvery} \mid \mathit{PosPredAny} \mid \mathit{PosPredEvery} \mid \\
&\quad \mathit{AlwISuccAny} \mid \mathit{AlwISuccEvery} \mid \mathit{PosISuccAny} \mid \mathit{PosISuccEvery} \mid \\
&\quad \mathit{AlwIPredAny} \mid \mathit{AlwIPredEvery} \mid \mathit{PosIPredAny} \mid \mathit{PosIPredEvery} \\
\mathit{AnyAll} &\triangleq \mathit{Any} \mid \mathit{All}
\end{aligned}
$$

A *Predicate* can consist of a simple *TaskPos*, with the intended semantics what the basic predicate *posoccur* specifies; a *TaskAlw*, with the intended semantics what the basic predicate *alwoccur* specifies; a *TaskRel*, with the intended semantics that all process models satisfying that particular relation should be retreived; or, it can be recursively defined as a binary or unary *Predicate* through the application of logical operators.

$$
\begin{aligned}
\mathit{Predicate} &\triangleq \mathit{TaskPos} \mid \mathit{TaskAlw} \mid \mathit{TaskRel} \mid \mathit{Bin\_Predicate} \mid \mathit{Un\_Predicate} \\
\mathit{Bin\_Predicate} &\triangleq o : \mathit{BinLogOp}; p_1, p_2 : \mathit{Predicate} \\
\mathit{Un\_Predicate} &\triangleq o : \mathit{UnLogOp}; p : \mathit{Predicate} \\
\mathit{BinLogOp} &\triangleq \mathit{And} \mid \mathit{Or} \\
\mathit{UnLogOp} &\triangleq \mathit{Not} \\
\mathit{TaskPos} &\triangleq l : \mathit{TaskLabel}; d : \mathit{SimDegree} \\
\mathit{TaskAlw} &\triangleq l : \mathit{TaskLabel}; d : \mathit{SimDegree}
\end{aligned}
$$

A *TaskRel* can be 1) a relationship between a *Task* and a *TaskSet* to check whether the *Task* occurs in the *TaskSet* (*TaskInTaskSet*), 2) a relationship between a *Task* and a *TaskSet* and involving a *TaskCompOp* and an *AnyAll* indicator to determine whether the *Task* has the *TaskCompOp* relationship with any/all *Tasks* in the *TaskSet* (*Task_TaskSet*), 3) a relationship between two *Tasks* involving a *TaskCompOp* predicate determining whether for the two *Tasks* that predicate holds (*Task_Task*), 4) a relationship between two *TaskSets* involving a *TaskCompOp* and an *AnyAll* indicator to determine whether the *Tasks* in those *TaskSets* all have the *TaskCompOp* relationship to each other or whether for each *Task* in the first *TaskSet* there is a corresponding *Task* in the second *TaskSet* for which the relationship holds (*Elt_TaskSet_TaskSet*), or 5) a relationship between two *TaskSets* determined by a set comparison operator (*Set_TaskSet_TaskSet*).

$$
\begin{aligned}
\mathit{TaskRel} &\triangleq \mathit{TaskInTaskSet} \mid \mathit{Task\_TaskSet} \mid \\
&\quad \mathit{Task\_Task} \mid \mathit{Elt\_TaskSet\_TaskSet} \mid \\
&\quad \mathit{Set\_TaskSet\_TaskSet} \\
\mathit{TaskInTaskSet} &\triangleq t : \mathit{Task}; ts : \mathit{TaskSet} \\
\mathit{Task\_TaskSet} &\triangleq t : \mathit{Task}; ts : \mathit{TaskSet}; \\
&\quad o : \mathit{TaskCompOp}; a : \mathit{AnyAll} \\
\mathit{Task\_Task} &\triangleq t_1, t_2 : \mathit{Task}; o : \mathit{TaskCompOp}
\end{aligned}
$$

$$
\begin{array}{rcl}
\mathit{Elt\_TaskSet\_TaskSet} & \triangleq & \mathit{ts}_1, \mathit{ts}_2 : \mathit{TaskSet}; o : \mathit{TaskCompOp}; \\
 & & a : \mathit{AnyAll} \\
\mathit{Set\_TaskSet\_TaskSet} & \triangleq & \mathit{ts}_1, \mathit{ts}_2 : \mathit{TaskSet}; o : \mathit{SetCompOp} \\
\mathit{SetCompOp} & \triangleq & \mathit{Identical} \mid \mathit{Subsetof} \mid \mathit{Overlap}
\end{array}
$$

## 3 The Semantics of APQL

We use denotational semantics to formally describe the semantics of APQL and adopt the notation in [5]. For each nonterminal $T$ we introduce a semantic function $M_T$ which defines the meaning of the nonterminal in terms of its parts.

First, we introduce some auxiliary notation in order to facilitate the subsequent definition of the semantics.

**Definition 1 (overriding union).** *The overriding union of $f : X \to Y$ by $g : X \to Y$, denoted as $f \oplus g$, is defined by $g \cup f \setminus \{(x, f(x)) \mid x \in dom(f) \cap dom(g)\}$.*

With the set of 20 basic predicates defined in the previous section, we use $\mathbb{BP}_u$ to denote the set of two *unary predicates* $\{posoccur, alwoccur\}$ which specify unary task relations, and similarly we use $\mathbb{BP}_b$ to denote the set of 18 *binary predicates* which specify binary task relations. The following two definitions introduce a higher order predicate that takes as input a unary or binary predicate, respectively. Note that the semantics of each predicate ($\phi/\psi$) is language independent. For a task $t$ in process model $N$, $L_N(t)$ specifies the label of $t$. A process model may have *silent tasks* which do not capture any task or activity in the process but are used for modelling purposes, e.g. a silent task used to capture an internal action that cannot observed by external uses. For a silent task $t$, we let $L(t) = \tau$.

**Definition 2.** *Let $N$ be a process model and $T$ the set of tasks in $N$, for $t_1, t_2 \in T$ and $\phi \in \mathbb{BP}_b$*

$$\mathsf{ref}^{\phi}(t_1, t_2, N) = \begin{cases} \phi(t_1, t_2, N) \textbf{ if } L_N(t_1) \neq \tau \wedge L_N(t_2) \neq \tau \\ \mathit{FALSE} \qquad \textbf{otherwise} \end{cases}$$

*i.e. the relation $\phi$ should hold between $t_1$ and $t_2$ in $N$ if both are non-silent tasks.*

**Definition 3.** *Let $N$ be a process model and $T$ the set of tasks in $N$, for $t \in T$ and $\psi \in \mathbb{BP}_u$*

$$\mathsf{ref}^{\psi}(t, N) = \begin{cases} \psi(t, N) \textbf{ if } L_N(t) \neq \tau \\ \mathit{FALSE} \textbf{ otherwise} \end{cases}$$

*i.e. the relation $\psi$ should hold for $t$ in $N$ if $t$ is a non-silent task.*

As queries may use variables, we must know their values during query evaluation. A *Binding* is an assignment of task sets to variables:

$$\mathit{Binding} \triangleq \mathit{ProcessModel} \times \mathit{Varname} \rightarrowtail 2^{\mathit{Task}}$$

Queries are applied to a repository of process models, i.e.

$$Repository \triangleq 2^{ProcessModel}$$

A process model $r$ consists of a collection of tasks $T_r$. For each task $t$ in process model $r$ we can retrieve its label as $L_r(t)$. Label similarity can be determined through the function $Sim$, where $Sim(l_1, l_2)$ determines the degree of similarity between labels $l_1$ and $l_2$ (which yields a value in the range [0,1]). Note that $Sim$ is a parameter of the approach in which case one can choose his/her own similarity notion and the corresponding function $Sim$ returns the similarity evaluation result to this parameter.

The query evaluation function $M_{Query}$ takes a query and a repository as input and yields those process models in that repository that satisfy the query:

$$M_{Query} : Query \times Repository \rightarrow 2^{ProcessModel}$$

This function is defined as follows:

$$M_{Query}[q : Query, R : Repository] \triangleq M_{Predicate}(q.p, R, M_{Assignments}(q.s, R, \varnothing))$$

The evaluation of the query evaluation function depends on the evaluation of the predicate involved and the assignments involved. When evaluating a sequence of assignments we have to remember the values that have been assigned to the variables involved. Inititally this set of assignments is empty.

$$M_{Assignments} : Assignments \times Repository \times Binding \rightarrow Binding$$

The result of a sequence of assignments is a binding where the variables used in the assignments are bound to sets of tasks. If a variable was already assigned a set of tasks in an earlier assignment in the sequence the latest assignment takes precedence over the earlier assignment.

$M_{Assignments}[s : Assignments, R : Repository, B : Binding] \quad \triangleq$
  **if** $\neg(s.TAIL).EMPTY$ **then**
    $M_{Assignments}(s.TAIL, R, B \oplus M_{Assignment}(s.FIRST, R, B))$
  **else** $B$

The result of an individual assignment is also a binding where the variable is linked to the set of tasks involved.

$M_{Assignment} : Assignment \times Repository \times Binding \rightarrow Binding$

$M_{Assignment}[a : Assignment, R : Repository, B : Binding)] \quad \triangleq$
  $\{((r, a.v), M_{TaskSet}(a.ts, R, B)(r)) \mid r \in R\}$

A predicate can be evaluated in the context of a repository and a binding and the result is a set of process models from that repository.

$$M_{Predicate} : Predicate \times Repository \times Binding \rightarrow 2^{ProcessModel}$$

A predicate may yield all process models in the repository that contain a task sufficiently similar to that task (with respect to the task label and similarity

degree). A predicate may also specify relationship between tasks (i.e. a *TaskRel*) in which case it yields all the process models that satisfy this relationship. A conjunction yields the intersection of the process models of the predicates involved, while a disjunction yields the union. The negation of a predicate yields the process models in the repository that do not satisfy the predicate.

$$
\begin{array}{l}
M_{Predicate}(p : Predicate, R : Repository, B : Binding) \quad \triangleq \\
\quad \textbf{case } p \textbf{ of} \\
\quad\quad TaskPos \Rightarrow \{r \in R \mid \exists t \in T_r[Sim(p.l, L_r(t)) \geq p.d \wedge posoccur(t, r)]\} \\
\quad\quad TaskAlw \Rightarrow \{r \in R \mid \exists t \in T_r[Sim(p.l, L_r(t)) \geq p.d \wedge alwoccur(t, r)] \\
\quad\quad TaskRel \Rightarrow M_{TaskRel}(p, R, B) \\
\quad\quad Bin\_Predicate \Rightarrow \\
\quad\quad\quad \textbf{case } p.o \textbf{ of} \\
\quad\quad\quad\quad And \Rightarrow M_{Predicate}(p.p_1, R, B) \cap M_{Predicate}(p.p_2, R, B) \\
\quad\quad\quad\quad Or \Rightarrow M_{Predicate}(p.p_1, R, B) \cup M_{Predicate}(p.p_2, R, B) \\
\quad\quad\quad \textbf{end} \\
\quad\quad Un\_Predicate \Rightarrow R \backslash M_{Predicate}(p, R, B) \\
\quad \textbf{end}
\end{array}
$$

A *TaskRel* in the context of a repository and a binding yields a set of process models in that repository.

$$M_{TaskRel} : TaskRel \times Repository \times Binding \rightarrow 2^{ProcessModel}$$

A *TaskRel* can be used to determine whether a task in a process model occurs in a given task set, whether a given basic predicate holds between a task in a process model and one or all tasks in a given task set, whether a given basic predicate holds between tasks in a process model, whether a given basic predicate holds between two or between all tasks in two given task sets, or whether a given set comparison relation holds between two given task sets.

$$
\begin{array}{l}
M_{TaskRel}(tr : TaskRel, R : Repository, B : Binding) \quad \triangleq \\
\quad \textbf{case } tr \textbf{ of} \\
\quad\quad TaskInTaskSet \Rightarrow \\
\quad\quad\quad \{r \in R \mid \exists v \in M_{TaskSet}(tr.ts, R, B)(r)[Sim(tr.t.l, L_r(v)) \geq tr.t.d]\} \\
\quad\quad Task\_TaskSet \Rightarrow \\
\quad\quad\quad \textbf{case } tr.a \textbf{ of} \\
\quad\quad\quad\quad Any \Rightarrow \{r \in R \mid \exists t_1 \in T_r \ \exists t_2 \in M_{TaskSet}(tr.ts, R, B)(r) \\
\quad\quad\quad\quad\quad [Sim(tr.t.l, L_r(t_1)) \geq tr.t.d \wedge rel^{tr.o}(t_1, t_2, r)]\} \\
\quad\quad\quad\quad All \Rightarrow \{r \in R \mid \exists t_1 \in T_r \ \forall t_2 \in M_{TaskSet}(tr.ts, R, B)(r) \\
\quad\quad\quad\quad\quad [Sim(tr.t.l, L_r(t_1)) \geq tr.t.d \wedge rel^{tr.o}(t_1, t_2, r)]\} \\
\quad\quad\quad \textbf{end} \\
\quad\quad Task\_Task \Rightarrow \{r \in R \mid \exists v_1, v_2 \in T_r[Sim(tr.t_1.l, L_r(v_1)) \geq tr.t_1.d \wedge \\
\quad\quad\quad\quad\quad Sim(tr.t_2.l, L_r(v_2)) \geq tr.t_2.d \wedge rel^{tr.o}(v_1, v_2, r)]\} \\
\quad\quad Elt\_TaskSet\_TaskSet \Rightarrow \\
\quad\quad\quad \textbf{case } tr.a \textbf{ of} \\
\quad\quad\quad\quad Any \Rightarrow \{r \in R \mid \exists t_1 \in M_{TaskSet}(tr.ts_1, R, B)(r) \\
\quad\quad\quad\quad\quad\quad \exists t_2 \in M_{TaskSet}(tr.ts_2, R, B)(r)[rel^{tr.o}(t_1, t_2, r)]\} \\
\quad\quad\quad\quad All \Rightarrow \{r \in R \mid \forall t_1 \in M_{TaskSet}(tr.ts_1, R, B)(r) \\
\quad\quad\quad\quad\quad\quad \forall t_2 \in M_{TaskSet}(tr.ts_2, R, B)(r)[rel^{tr.o}(t_1, t_2, r)]\} \\
\quad\quad\quad \textbf{end}
\end{array}
$$

*Set_TaskSet_TaskSet* ⇒
**case** *tr.o* **of**
*Identical* ⇒
$\{r \in R \mid M_{TaskSet}(tr.ts_1, R, B)(r) = M_{TaskSet}(tr.ts_2, R, B)(r)\}$
*Subsetof* ⇒
$\{r \in R \mid M_{TaskSet}(tr.ts_1, R, B)(r) \subseteq M_{TaskSet}(tr.ts_2, R, B)(r)\}$
*Overlap* ⇒
$\{r \in R \mid M_{TaskSet}(tr.ts_1, R, B)(r) \cap M_{TaskSet}(tr.ts_2, R, B)(r) \neq \varnothing\}$
**end**
**end**

A *TaskSet* within the context of a repository and a binding yields a mapping which assigns to each process model in the repository the collection of tasks within that model that satisfy the restriction imposed by the *TaskSet*.

$$M_{TaskSet} : TaskSet \times Repository \times Binding \rightarrow (ProcessModel \rightarrow 2^{Task})$$

When a *TaskSet* is a set of tasks, then for each process model the result is the set of tasks within that process model that are sufficiently similar to at least one of the tasks in that *TaskSet*. When the *TaskSet* is a variable, then the evaluation is similar except that the task set used is the task set currently bound to that variable. *TaskSets* can also be formed through *Construction* (where the set operators union, difference, and intersection are used) or *Application* (where task sets are formed through set comprehension, i.e. they are defined through properties that they have - these properties relate to the basic predicates).

$M_{TaskSet}(tks : TaskSet, R : Repository, B : Binding) \quad \triangleq$
**case** *tks* **of**
*SetofTasks* ⇒
$\{(r, \{t \in T_r \mid \exists_{1 \leq i \leq tks.LENGTH}[Sim(tks(i).l, L_r(t)) \geq tks(i).d]\}) \mid r \in R\}$
*TaskSetVar* ⇒
$\{(r, X) \mid r \in R\}$ **where**
$X = \begin{cases} B(r, tks) & \textbf{if } (r, tks) \in dom(B) \\ \varnothing & \textbf{otherwise} \end{cases}$
*Construction* ⇒
**case** *tks.o* **of**
*Union* ⇒
$\{(r, M_{TaskSet}(tks.ts_1, R, B)(r) \cup M_{TaskSet}(tks.ts_2, R, B)(r)) \mid r \in R\}$
*Difference* ⇒
$\{(r, M_{TaskSet}(tks.ts_1, R, B)(r) \backslash M_{TaskSet}(tks.ts_2, R, B)(r)) \mid r \in R\}$
*Intersection* ⇒
$\{(r, M_{TaskSet}(tks.ts_1, R, B)(r) \cap M_{TaskSet}(tks.ts_2, R, B)(r)) \mid r \in R\}$
**end**
*Application* ⇒
**case** *tks.a* **of**
*Any* ⇒
$\{(r, \{t \in T_r \mid \exists v \in M_{TaskSet}(tks.ts, R, B)(r)[rel^{tks.o}(t, v, r)]\}) \mid r \in R\}$
*All* ⇒
$\{(r, \{t \in T_r \mid \forall v \in M_{TaskSet}(tks.ts, R, B)(r)[rel^{tks.o}(t, v, r)]\}) \mid r \in R\}$
**end**
**end**

## 4 Examples of APQL Queries

In this section we present some sample queries and show how they can be captured in APQL in order to further illustrate the language. The sample queries, specified in natural language, are listed below (which are numbered $Q_1$ to $Q_{10}$). In these queries, by default the value for the *AnyAll* identifier, when applicable, is *all*, and by default the value for the SimDegree is 1. According to the abstract syntax of APQL in Sect. 2, Fig. 2 shows the grammar trees for queries $Q_1$ to $Q_6$, and Fig. 3 shows the grammar trees for queries $Q_7$ to $Q_{10}$. Note that in the following A to L are task labels (i.e. activity names).

$Q_1$. Select all process models where task A occurs in some process execution and task B occurs in every process execution.
$Q_2$. Select all process models where in every process execution it is possible that task A occurs before task D.
$Q_3$. Select all process models where in every process execution task A always occurs before task D.
$Q_4$. Select all process models where in some process execution it is possible that task A occurs before task B and task B occurs before task K.
$Q_5$. Select all process models where in some process execution task A always occurs before task B.
$Q_6$. Select all process models where task B occurs in parallel with task C.
$Q_7$. Select all process models where task B occurs in parallel with task C and where task A occurs in parallel with task H.
$Q_8$. Select all process models where in every process execution task B and task C never occur together.
$Q_9$. Select all process models where in some process execution the immediate predecessors of task H are among the immediate successors of task B.
$Q_{10}$. Select all process models where in some process execution the immediate predecessors of task H may occur after the common immediate successors of task B and task C.

In order to illustrate the formal semantics of APQL, a number of process models, represented in BPMN, are presented in Fig. 4. For each sample query above and for each model it is indicated whether the model is part of the answer to the query (in that case the box corresponding to the query is ticked otherwise the box is not ticked). Note that in some models tasks with the same label occur (e.g. there are two tasks labeled A in model (5) in Fig. 4), in which case, APQL will treat these tasks as same tasks during query evaluation[1].

## 5 Related Work

Mindful of the importance of query languages for business process models, the Business Process Management Initiative (BPMI) proposed to define a standard

[1] Note that APQL is query language rather than a process modelling language.

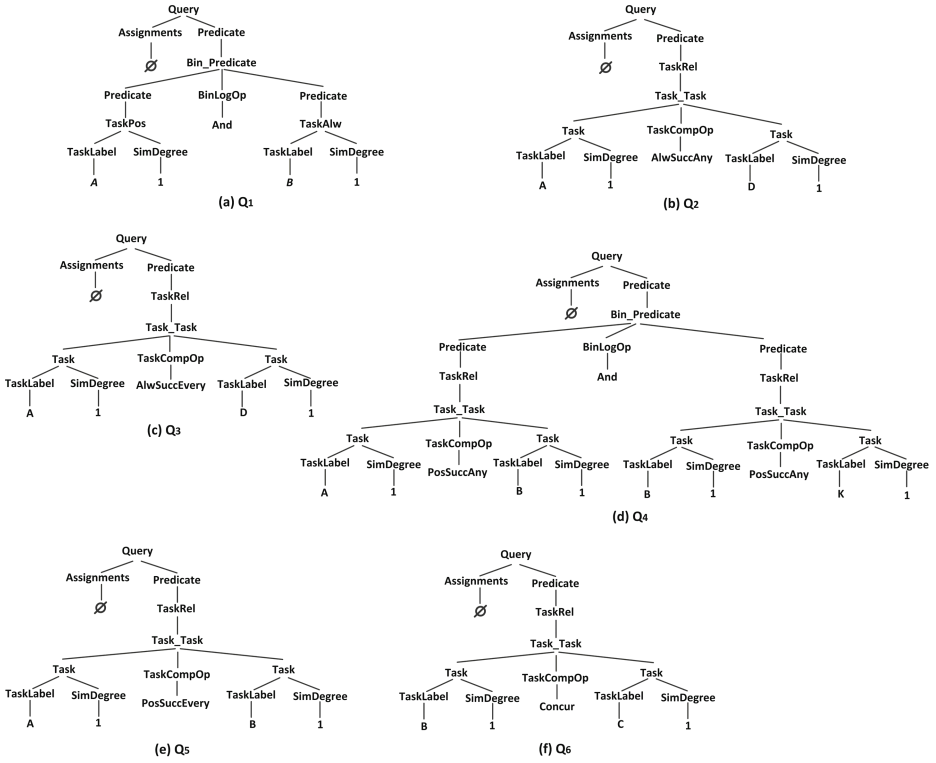
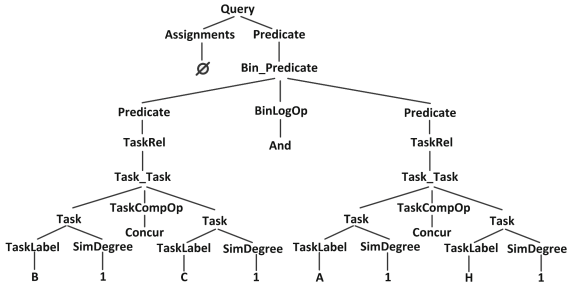


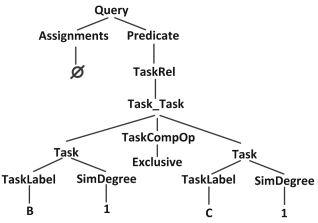

**Fig. 2.** The APQL grammar trees of sample queries $Q_1 - Q_6$

business process model query language in 2004[2]. While such a standard has never been published, two major research efforts have been dedicated to the development of query languages for process models. One is known as BP-QL [3], a visual query language based on an abstract representation of BPEL and supported by a formal model of graph grammars for processing of queries. BP-QL can be used to query process specifications written in BPEL rather than possible executions, and ignores the run-time semantics of certain BPEL constructs such as conditional execution and parallel execution. The other effort, namely BPMN-Q [1,6], is also a visual query language which extends a subset of the BPMN modelling notation and supports graph-based query processing. Similarly to BP-QL, BPMN-Q only captures the structural (i.e., syntactical) relations between tasks, and not their behavioral relationships. In [7], the authors explore the use of an information retrieval technique to derive similarities of activity names, and develop an ontological expansion of BPMN-Q to tackle the problem of querying business processes that are developed with different terminologies. A framework of tool support for querying process model repositories using BPMN-Q and its extensions is presented in [8]. Recently, in [9], an approach that applies

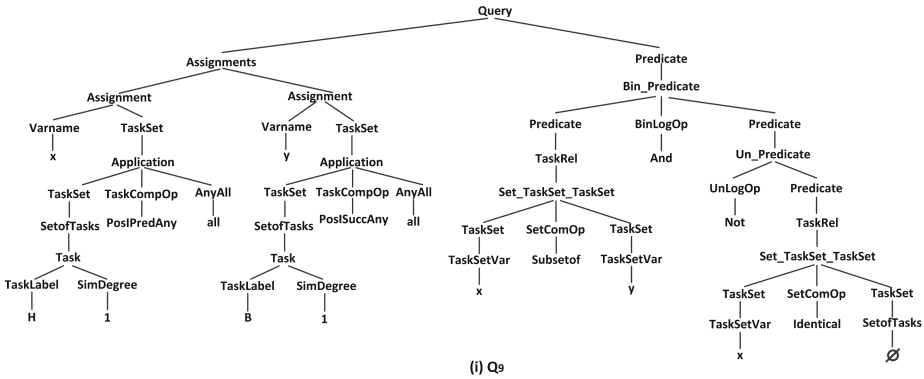
[2] `http://www.bpmi.org/downloads/BPMI_Phase_2.pdf`



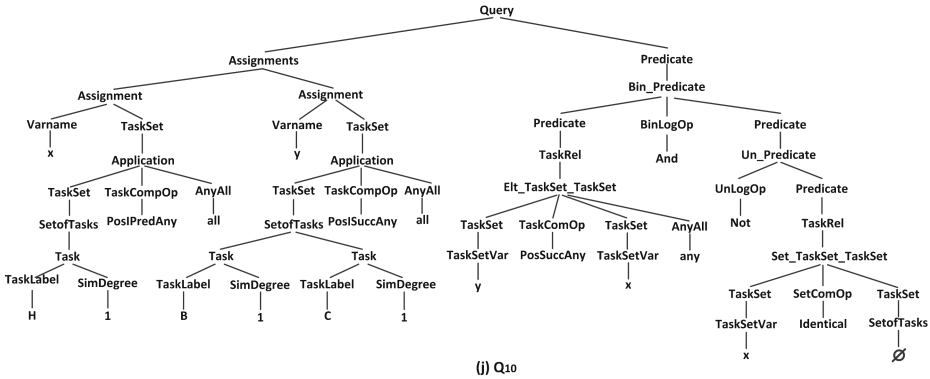

(g) $Q_7$




(h) $Q_8$




(i) $Q_9$




(j) $Q_{10}$

**Fig. 3.** The APQL grammar trees of sample queries $Q_7 - Q_{10}$

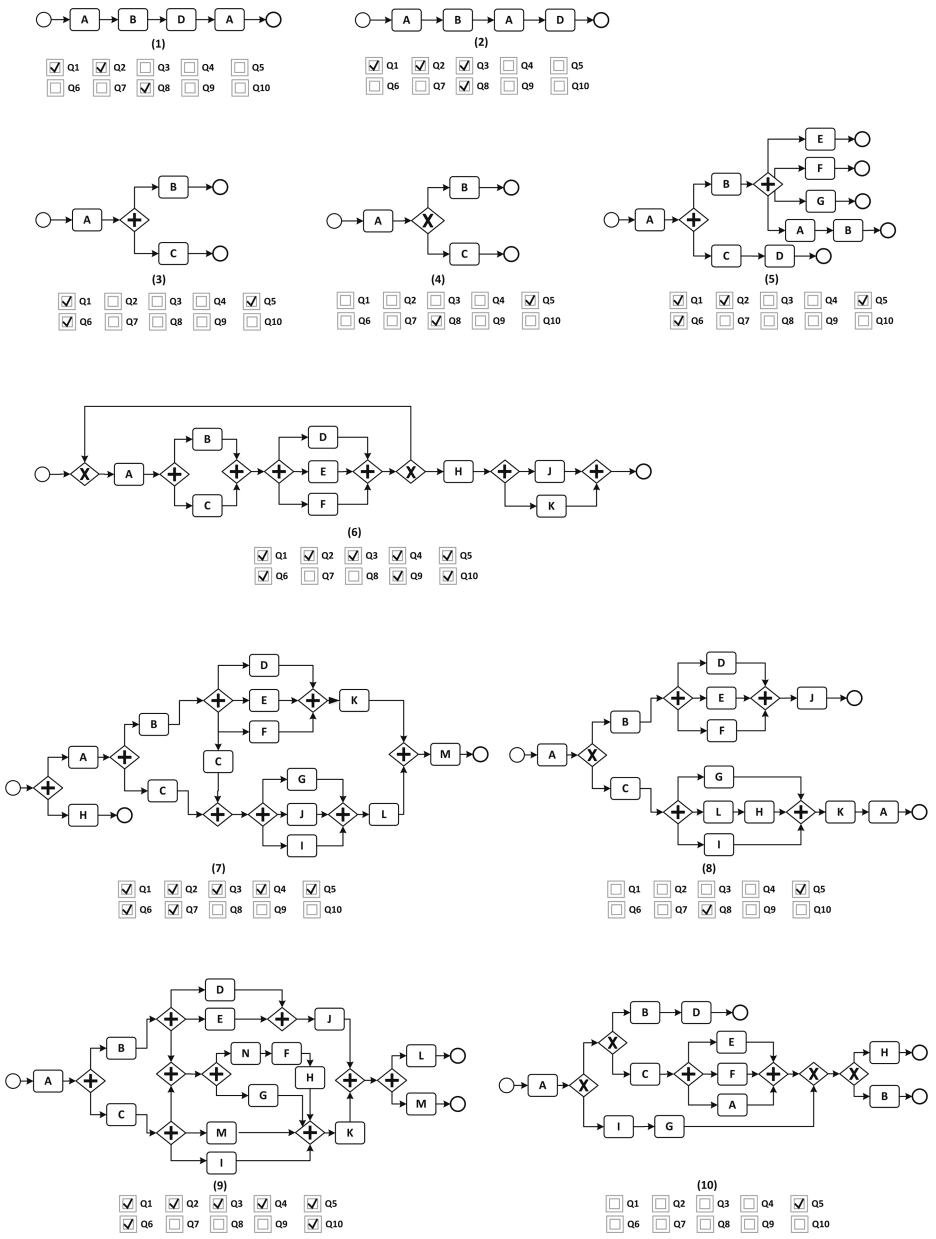



**Fig. 4.** A list of BPMN business process models and evaluation of sample queries $Q_1 - Q_{10}$ over these process models

an indexing method based on the (graph-based) flow relation between tasks in BPMN process diagrams is proposed for efficient processing of BPMN-Q queries.

APQL presents three distinguishing features compared to the above languages. First, its abstract syntax and semantics have been purposefully defined to be independent of a specific process modelling language (such as BPEL or BPMN). This will allow APQL and its query evaluation technique to be implemented for a variety of process modelling languages. Second, APQL can express all possible temporal-ordering relations (precedence/succession, concurrence and exclusivity) between individual tasks, between an individual task and a set of tasks as well as between different sets of tasks. Third, APQL querying constructs need to be evaluated over the execution semantics of process models, rather than their structural relations. In fact, structural characteristics alone are not able to capture all possible order relations among tasks which can occur during execution, in particular with respect to cycles and task occurrences.

In addition to the development of process model query languages, other techniques are available in the literature which can be useful for querying process model repositories. In [10,11], the authors focus on querying the content of business process models based on metadata search. The VisTrails system [12] allows users to query scientific workflows by example and to refine workflows by analogies. WISE [13] is a workflow information search engine which supports keyword search on workflow hierarchies. In [14], the authors use graph reduction techniques to find a match to the query graph in the process graph for querying process variants, and the approach only works on acyclic graphs. In [15,16,17], a group of similarity-based techniques have been proposed which can be used to support process querying. In [18], a technique to query process model repositories is proposed based on an input Petri net. Finally, in [19], the notion of behavioural profile of a process model is defined, which captures dedicated behavioural relations like exclusiveness or potential occurrence of activities. However, these behavioural relations are derived from the structure of a process model. Thus, for the reasons mentioned above, behavioral profiles only provide an approximation of a process model's behavior, whereas APQL can precisely determine whether or not a process model satisfies a given query.

## 6 Conclusions

This paper contributes an innovative language, namely APQL, for querying process model repositories. APQL provides three main advantages over the state of the art. First, the language is expressive since it allows users to specify all possible order relationships among tasks or sets thereof. Second, the language is precise, since APQL queries are defined for evaluation over process model behavior, while existing query languages only support structural process characteristics. Third, the language's syntax and semantics are defined independently of any specific process modeling language.

The next stage is to operationalise APQL and the main task is to develop a technique for evaluation of APQL queries. This evaluation technique could be designed on top of a well established mathematical technique for describing

behavioural semantics, e.g. Petri nets. One challenge though, when it comes to determining semantic relationships between tasks, is how to determine these relationships in a feasible manner (i.e. without suffering from the well-known state space explosion problem).

Currently APQL only focuses on the control flow perspective of business process models. In the future, we will extend the language definition in order to include other process perspectives such as data and participating resources. Moreover, we plan to run structured interviews with domain experts to assess the overall ease of use and usefulness of APQL.

**Acknowledgements.** Song and Wang are supported by the National Basic Research Program of China (2009CB320700), the National High-Tech Development Program of China (2008AA042301), the Project of National Natural Science Foundation of China (90718010), and the Program for New Century Excellent Talents in University of China. ter Hofstede, La Rosa and Polyvyanyy are partly supported by the ARC Linkage Grant "Facilitating Business Process Standardization and Reuse" (LP110100252). In 2010 and 2011, ter Hofstede was a senior visiting scholar of Tsinghua University. NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

## References

1. Awad, A.: BPMN-Q: A language to query business processes. In: Proceedings of the 2nd International Workshop on Enterprise Modelling and Information Systems Architectures (EMISA 2007). LNI, vol. P-119, pp. 115–128. GI (2007)
2. Beeri, C., Eyal, A., Kamenkovich, S., Milo, T.: Querying business processes. In: Proceedings of the 32nd International Conference on Very Large Data Bases, pp. 343–354. ACM (2006)
3. Beeri, C., Eyal, A., Kamenkovich, S., Milo, T.: Querying business processes with BP-QL. Inf. Syst. 33(6), 477–507 (2008)
4. OMG: Business Process Model and Notation (BPMN) version 2.0 (January 2011), `http://www.omg.org/spec/BPMN/2.0`
5. Meyer, B.: Introduction to the Theory of Programming Languages. Prentice-Hall (1990)
6. Awad, A., Decker, G., Weske, M.: Efficient compliance checking using BPMN-Q and temporal logic. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240, pp. 326–341. Springer, Heidelberg (2008)
7. Awad, A., Polyvyanyy, A., Weske, M.: Semantic querying of business process models. In: Proceedings of the 12th International IEEE Enterprise Distributed Object Computing Conference, pp. 85–94. IEEE Computer Society (2008)
8. Sakr, S., Awad, A.: A framework for querying graph-based business process models. In: Proceedings of the 19th International Conference on World Wide Web, pp. 1297–1300. ACM (2010)
9. Awad, A., Sakrb, S.: On efficient processing of BPMN-Q queries. Computers in Industry 63(9), 867–881 (2012)

10. Vanhatalo, J., Koehler, J., Leymann, F.: Repository for business processes and arbitrary associated metadata. In: BPM 2006. LNCS, vol. 4102, pp. 426–431. Springer (2006)
11. Wasser, A., Lincoln, M., Karni, R.: ProcessGene Query – a tool for querying the content layer of business process models. In: BPM 2006. LNCS, vol. 4102, pp. 1–8. Springer (2006)
12. Scheidegger, C.E., Vo, H.T., Koop, D., Freire, J., Silva, C.T.: Querying and reusing workflows with VisTrails. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 1251–1254. ACM (2008)
13. Shao, Q., Sun, P., Chen, Y.: WISE: A workflow information search engine. In: Proceedings of the 25th International Conference on Data Engineering, pp. 1491–1494. IEEE Computer Society (2009)
14. Lu, R., Sadiq, S.W.: Managing process variants as an information resource. In: Dustdar, S., Fiadeiro, J.L., Sheth, A.P. (eds.) BPM 2006. LNCS, vol. 4102, pp. 426–431. Springer, Heidelberg (2006)
15. van der Aalst, W.M.P., de Medeiros, A.K.A., Weijters, A.J.M.M.: Process equivalence: Comparing two process models based on observed behavior. In: Dustdar, S., Fiadeiro, J.L., Sheth, A.P. (eds.) BPM 2006. LNCS, vol. 4102, pp. 129–144. Springer, Heidelberg (2006)
16. Ehrig, M., Koschmider, A., Oberweis, A.: Measuring similarity between semantic business process models. In: Proceedings of the 4th Asia-Pacific Conference on Conceptual Modelling. CRPIT, ACS, vol. 67, pp. 71–80 (2007)
17. van Dongen, B.F., Dijkman, R., Mendling, J.: Measuring similarity between business process models. In: Bellahsène, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 450–464. Springer, Heidelberg (2008)
18. Jin, T., Wang, J., Wu, N., La Rosa, M., ter Hofstede, A.H.M.: Efficient and accurate retrieval of business process models through indexing (short paper). In: Meersman, R., Dillon, T.S., Herrero, P. (eds.) OTM 2010. LNCS, vol. 6426, pp. 402–409. Springer, Heidelberg (2010)
19. Weidlich, M., Mendling, J., Weske, M.: Efficient consistency measurement based on behavioral profiles of process models. IEEE Transactions on Software Engineering 37(3), 410–429 (2011)

# BPEL Similarity — A Metric Based on Activity Constraint Graphs

Jianchun Xing[1], Xuewei Zhang[1], Wei Song[2], Qiliang Yang[1], Jidong Ge[3], and Hongda Wang[1]

[1] PLA University of Science and Technology, Nanjing, China
[2] Nanjing University of Science and Technology, Nanjing, China
[3] Nanjing University, Nanjing, China
xueweizhang8877@gmail.com

**Abstract.** As the increasing popularity of Web Service Business Process Execution Language (WS-BPEL), it is urgent to meet the demand of retrieving the related BPEL processes in BPEL process repository quickly for business personnel. BPEL similarity retrieval technology is one of research focuses in the field of BPEL repository management system. As existing approaches tend to lack metric features and use structural aspects of BPEL processes rather than their behaviors, they are often not applicable for effective similarity search. In this paper, we propose a metric based on BPEL activity constraint graphs (BACGs) to calculate the similarity degree of BPEL processes. It is grounded on the Jaccard coefficient and leverages behavioral relations between BPEL activities. The metric is successfully evaluated towards its approximation of human similarity assessment.

**Keywords:** BPEL Process, Similarity, BPEL Activity Constraint Graph, Behavioral Metric.

## 1 Introduction

With the growth of Web services market share, as a Web service orchestration language, BPEL [1] is gradually becoming the mainstream. BPEL is formulated as a specification of Web services composition to address the heterogeneity of language, platform, protocol and data. More and more enterprises and scientific research teams have built the large-scale BPEL process repositories. Such repositories may contain hundreds or even thousands of business processes, e.g., we have accessed to a repository of the Oracle Company containing nearly 300 BPEL processes. An important issue arising from BPEL process applications is how to retrieve BPEL processes from the expanding BPEL repositories conveniently, accurately and effectively. Actually, the capability to easily retrieve useful BPEL processes becomes increasingly critical in several situations, e.g., when adding a new BPEL process into a BPEL repository, it is necessary to search the repository whether there are similar BPEL processes; in the context of company mergers, in order to analyze the overlap of BPEL processes and identify areas for consolidation, the process analysts need to identify common or similar processes between the merged companies; in some system that could not tolerate any

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-319-02922-1_10

M. Song, M.T. Wynn, and J. Liu (Eds.): AP-BPM 2013, LNBIP 159, pp. 39–55, 2013.

failure BPEL processes, we must find a similar one to replace the failure one; given the as-is and to-be BPEL processes, we would like to know how much they differ from each other and how we can efficiently transform the as-is to the to-be process. Unfortunately, there is no available and effective BPEL retrieval technology to meet the user's demands. Traditional database retrieval technology is not suitable, because it cannot fitly sort and obtain the useful hash for BPEL processes. Text-based search and folder navigation technologies are too rough to efficiently find the desirable BPEL processes [2]. The heterogeneity of BPEL processes leads exact pattern matches to being useless. Therefore, studying the similarity between BPEL processes has been imminent.

At present, the majority of existing works focuses on textual similarity and structural similarity. Various notions of edit distance are used to this end [3-5]. However, those approaches may not adequately take the behavior of an executable BPEL process into account, as it is pointed out in [6] that two processes may look quite similar, considering the activity labels and the process structure, but may behave quite differently. Then, many behavioral similarity measures have been proposed [7-9]. Although these measures may work well in some aspects, there are some drawbacks for determining the similarity of BPEL processes. First, much detailed information arises for an executable BPEL process, e.g. data information, partnerLinks. They are not applicable to accurately measure the similarity degree between BPEL processes for those approaches from the field of process model, because the important information has been abstracted away in process models. Second, the majority of those measures [5, 10] cannot satisfy the triangle inequality principle [11, 12]. This means that every search must carry on the exhaustive search. It leads to very low search efficiency for the management, maintenance and retrieval of the large-scale BPEL repositories.

In order to address these problems, we propose a behavioral metric that quantifies the similarity between BPEL processes. It enables effective similarity search since it satisfies metric properties, in particular the triangle inequality. The metric builds on BPEL Activity Constraint Graph (BACG), an abstraction of the behavior of a BPEL process. These BACGs capture behavioral constraints between every activity pair, e.g., partial order relation, mutual-exclusion relation. Using this abstraction, we propose five elementary similarity measures. Based on these measures, we construct a behavioral metric that quantifies the similarity of BPEL processes. As an evaluation, we conducted experiments using the examples from Oracle BPEL Process Manager Samples. Compared with manual similarity judgments by the BPEL process experts, our metric shows a good approximation of human similarity assessment.

The key contribution of our work is threefold: first, a novel quantitative approach is proposed to quantify the similarity of BPEL processes by leveraging activity constraints. Second, our approach is less sensitive to some observable behavioral relations, e.g., strict order relation, flow relation, but it can capture the slight alterations of some essential activity constraints between two activities from a BPEL process, e.g., partial order relation. Third, in the field of BPEL process, we apply our approach by analyzing the similarity between some real-life executable BPEL processes.

The remainder of the paper is structured as follows. Section 2 gives the basic concepts used throughout this paper. Our metric to quantify the similarity between BPEL processes is introduced in Section 3. Section 4 is concerned with its evaluation. We discuss the limitations of our approach in Section 5. Finally, Section 6 and Section 7 present related works and the conclusion.

## 2 Preliminaries

Search involves the query in comparing with objects from a repository. The complexity of every comparison algorithm and the size of repository determine the efficiency of the search. We realize that the complexity is equal to the complexity of every comparison algorithm multiplied by the number of comparison operations. If we can reduce the number of comparison operations, it will be overall. Those methods based on tree or hash index are not applicable in a BPEL repository, because BPEL process cannot be mapped to the data structure effectively and the result of mapping also is unordered and ataxonomic. Therefore, industry and academy solve the problems by using the properties of distance function [13].

**Definition 1 (Metric)** [9]. *A metric is a distance function d:* $X \times X \rightarrow \mathbb{R}$ *between objects of domain X with the following properties*:

- *Symmetry*: $\forall p_m, p_n \in X$, then $d(p_m, p_n) = d(p_n, p_m)$
- *Nonnegativity*: $\forall p_m, p_n \in X, p_m \neq p_n$, then $d(p_m, p_n) > 0$
- *Identity*: $\forall p_m, p_n \in X$, then $d(p_m, p_n) = 0 \Leftrightarrow p_m = p_n$
- *Triangle inequality*: $\forall p_m, p_n, p_l \in X$, then $d(p_m, p_l) < d(p_m, p_n) + d(p_n, p_l)$

*A metric space is a pair* $S = (X, d)$.

The triangle inequality enables to determine minimum and maximum distances of the two objects $p_m$, $p_n$ without calculating it, given their pairwise distances to a third object $p_l$. In order to search efficiently in a large-scale BPEL repository, i.e., avoid comparison with every process object, the repository is split into partitions which some can be pruned during search. In metric spaces, a partition which is established by a pivot $o \in X$ and a covering radius $r(o) \in \mathbb{R}$, spans a sphere around $o$. All BPEL process objects that their distances with the pivot $o$ are less than the radius $r(o)$ belong to the sphere $U(o)$. Then, similarity search requires a query process $q \in X$ from the same domain of the process objects in a repository, and a tolerance, i.e., a query radius $r(q) \in \mathbb{R}$, expresses how similar the process objects may be to $q$. Those process

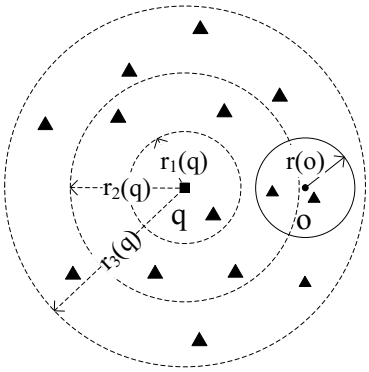



**Fig. 1.** Metric space partition $U(o)$ (solid circle) with pivot $o$ and similarity query (dotted concentric circle) with a query process $q$

objects that the similarity degrees are greater than the $r(q)$ of the query process can be returned in the sphere $U(q)$ of the query process. For the sphere $U(o)$ of the query process $o$ and the sphere $U(q)$ of the process object $q$, there are three following situations on the metric space — exclusion, inclusion and intersection [9][14], cf. Fig.1:

- Exclusion — $r(o) + r_1(q) < d(o, q)$: All objects in $U(o)$ are further away from $q$ than $r_1(q)$, i.e., they cannot be returned by the retrieval. The distance from $q$ to any other object in $U(o)$ does not have to be calculated.
- Intersection — $r(o) + r_2(q) \geq d(o, q)$: This means the identification of the objects in the intersection to require exhaustive search of $U(o)$.
- Inclusion — $r(o) - r_3(q) \leq d(o, q)$: $U(o)$ is completely included in the sphere around $q$. Thus, all objects satisfy the similarity constraint and need not be compared with the query unless a ranking of the search result is desired.

## 3 Behavioral Similarity Metrics

In this section, we present our approach which measures the similarity between BPEL processes based on activity constraints and Jaccard coefficient. We first give the definition of BACG in Section 3.1 and analyze the similarity measurement of BPEL processes based on BACGs in Section 3.2. Then, five elementary similarity metrics are shown in Section 3.3. Finally, we define aggregated metric for BACGs in Section 3.4.

### 3.1 BPEL Activity Constraint Graph

In order to precisely capture the behavioral relations between an activity pair from a BPEL process, we propose the notion of BACG which provides the foundation to reason about similarity of a pair. The following definition of BACG is shown.

**Definition 2 (BPEL Activity Constraint Graph, BACG).** *A BPEL activity constraint graph is a graph <N, E>, where*

- *N is a set of activities that contains one Entry activity, basic activities and structured activities.*
- *$E = E_1 \cup E_2 \subseteq N \times N$ is a set of edges. A directed edge $<x_1, y_1> \in E_1$ denotes a partial order relation and an undirected edge $(x_2, y_2) \in E_2$ denotes a mutual-exclusion relation.*
  - *— Partial orders refer to control dependence, data dependence and asyn-invocation dependence.*
  - *— Mutual-exclusion refers to the requirement of ensuring that only one activity in the activity pair can be performed at a time.*

A control dependence [15] indicates that an activity determines whether or not another activity can be executed. A data dependence [15] indicates that an activity uses a variable that is defined by another activity. Data dependences can be classified into three categories in BPEL process: true-data dependence, anti-data dependence and output-data dependence. The anti-data and output-data dependences can be avoided through the variables renaming, so only the true-data dependence is considered in the

paper. An asyn-invocation dependence [16, 17] indicates that a <receive> activity is responsible for receiving the response of a one-way <invoke> activity. This dependence is caused by the asynchronous communicating mechanism between BPEL processes. In a BPEL process, when there are some choice activities, i.e., <if>, <switch>, <while>, since only a branch might be performed for this kind of structured activities at the same time, there is a mutual-exclusion relation between any activity pair from two different branches.

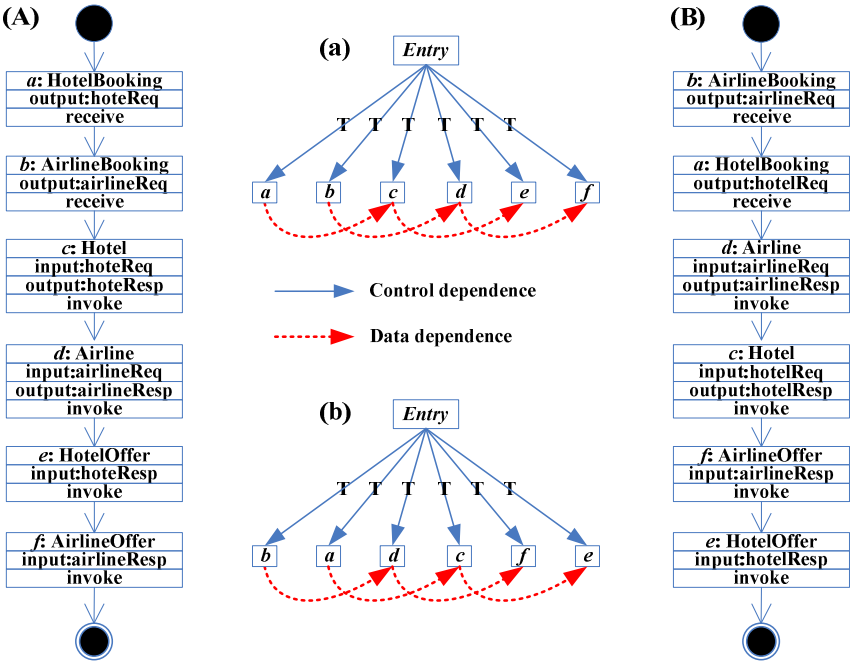



**Fig. 2.** The similarity degree is low based on behavioral profiles, but the two processes are very similar based on BACGs

In order to accurately quantify the similarity of the executable BPEL processes, the activity constraints of all activity pairs from each process should be considered. However, in BACG, there is no partial order relation or mutual-exclusion relation for the majority of the activity pairs. In this situation, we assume that there is an independence relation for those pairs. Actually, an independence relation can help to determine the differences between two BPEL processes, e.g., in a BPEL process the absence of an activity or introducing a new activity. To conveniently define the similarity measures later, for a BPEL process $P(A)$ ($A$ is a set of activities that contains basic activities and structure activities), we use the notations to denote activity constraints of the activity pairs, i.e., $\{\rightarrow_P$ (control dependence), $>_P$ (data dependence), $\downarrow_P$ (asyn-invocation), $+_P$ (mutual-exclusion), $\circ_P$ (independence relation)$\}$. We realize that mutual-exclusion relation and independence relation belong to a total order

relation. As a universal set of the activity pairs from a BPEL process is considered, an inverse partial order relation is introduced in the inverse pair $(y, x)$ when there is a partial order relation between a pair $(x, y)$. However, an exceptional case is that we cannot consider the inverse data dependence — an activity $x$ is data dependent on an activity $y$ and $y$ is also data dependent on $x$. In addition, there is an independence relation between an activity and itself, if the activity is not data dependent on itself.

The authors of [18] proposed the notion of behavioral profile that defines three behavioral constraints between activities in process models, i.e. strict order relation, interleaving relation and exclusiveness relation. The approach that is applied in the field of process models is very excellent [9]. As we know, BPEL process is executable and it contains much specific information, e.g., partnerLink, the input and output of activity. Unfortunately, the important information has been abstracted away in process models, thus, behavioral profiles are not suitable to quantify the similarity of BPEL processes. Compared with behavioral profile, our approach can capture the more essential behavioral constraints between activities from a BPEL process e.g., in Fig.2 (A) and (B), they are changed for the execution orders of those corresponding activities without partial order relations in BPEL processes (A) and (B). Based on behavioral profiles [9], this may heavily weaken the similarity degree of the two BPEL processes. However, as the corresponding BACGs (a) and (b) of BPEL processes (A) and (B) are consistent, this may not affect the similarity in our approach.

### 3.2 Analysis of Similarity Based on BACGs

Similarity measurement based on BACGs requires matching activities. Given two BPEL processes, the correspondences between activities have to be determined. These correspondences are used to measure the overlap of behavior in the two processes. In the paper, we consider that the name label of each activity is unique, and a corresponding is built only if some attributes between activities are the same. These attributes contain name, type, partnerLink and portType. Our work does not focus on this aspect of similarity, but rather on behavioral properties. Therefore, we assume these correspondences to be given, hereafter. To keep the formalization of our metrics concise, we abstract from such correspondences and assume corresponding activities to be identical. In other words, given two BPEL processes $X$ and $Y$ with their sets of activities $A_X$ and $A_Y$, a correspondence between an activity from $X$ and an activity from $Y$ is manifested as the existence of an activity $x \in (A_X \cap A_Y)$. In this paper, in order to avoid the rapid increase of activity constraints and the time complexity, we assume that there is only 1 : 1 correspondence between activities.

In order to measure the similarity of BPEL processes, we first transformed two BPEL processes into BPEL Control Flow Graphs (BCFGs) [19] in which each activity is associated with the input and output of a corresponding BPEL activity and its necessary attributes. Then, BACGs of the two BPEL processes are derived by analyzing the partial order relations and mutual-exclusion relations based on BCFGs. Note that in this paper we introduce an independence relation between an activity pair where the partial order relation and mutual-exclusion relation do not exist. Finally, we can calculate the similarity degree of the two BPEL processes based on Jaccard coefficient and the obtained BACGs (cf. Fig.3).

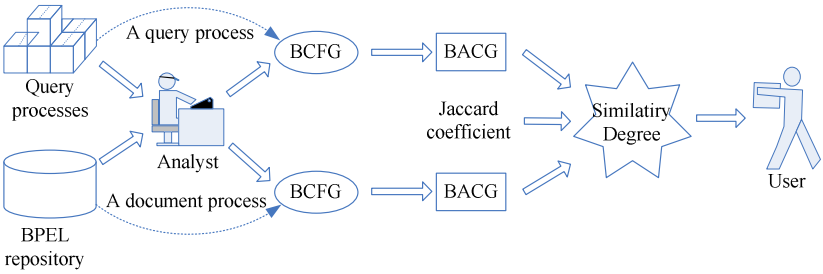



**Fig. 3.** The framework of our similarity analysis approach

### 3.3 Elementary Similarity Measurement

A BPEL process $X$ resembles another BPEL process $Y$ in certain behavioral aspects if they overlap in their BACGs $B_X$ and $B_Y$, respectively. The larger this overlap is, the more similar we assume two processes to be. We quantify the similarity by the well-known Jaccard coefficient. It is a static statistical formula used to measure the similarity and the degree of differences of two collections: $sim(A, B) = |A \cap B| / |A \cup B|$. First, since each kind of activity constraints is essentially a set in a BACG, this measure can easily be applied to BACGs. Second, it can be translated into a metric $d(A, B) = 1 - sim(A, B)$ [11] to enable effective similarity search.

According to the travel agency process in [17, 20], we make some adjustments and obtain two BPEL processes to clearly illustrate our problems (cf. Fig.4). The case is universal, since there are the characteristics of general BPEL processes for them. It shows two order handling processes ($M$, $N$), in which we should select a befitting hotel to book. The case has the characteristics of the general BPEL process as it is applied to many publications [16, 17]. The correspondences between activities are showed by equal labels in two processes, and are represented through the indices $a$ to $h$. Based on BCFGs and data information, their BACGs $B_M$ and $B_N$ are also obtained and activity constraints are depicted as matrices over the activities (cf. Fig.5).

As described previously, the partial order relation contains three types of dependences in a BACG. Since each type of dependences may arouse an important influence for measuring the similarity degree between two BPEL processes, each one should be defined as a similarity measurement which the inverse partial order relation should be introduced as depicted in Section 3.1. This can make sure that we quantify the similarity of BPEL processes more comprehensively and accurately from the perspective of the universal set of activity pairs. The corresponding similarity quantifies how many common pairs exist in two BPEL processes that feature the same activity constraints. Then, the five similarity measures of activity constraints are following:

**Definition 3 (Control Dependence Similarity — CDS).** *Let $X$, $Y$ be BPEL processes and $\rightarrow X$, $\rightarrow Y$ the control dependences, $\rightarrow X^{-1}$, $\rightarrow Y^{-1}$ the inverse control dependences of their respective BACGs ($B_X$, $B_Y$). Control Dependence Similarity is defined as*:

$$sim_{\rightarrow}(B_X, B_Y) = \frac{\left|(\rightarrow X \cup \rightarrow X^{-1}) \cap (\rightarrow Y \cup \rightarrow Y^{-1})\right|}{\left|(\rightarrow X \cup \rightarrow X^{-1}) \cup (\rightarrow Y \cup \rightarrow Y^{-1})\right|}.$$

We can find that there are the control dependences between the pairs (*b*, *c*), (*b*, *e*) in BPEL processes of Fig.5. The only difference between *M* and *N* with regard to the control dependence is that *d* and *f* are out of the structured activity *b* of *N*. It leads to the control dependences of two activity pairs (*b, d*) and (*b, f*), as well the inverse control dependences of their inverse pairs in *M* being absent in *N*. This yields a similarity $sim_{\rightarrow}(B_M, B_N) = 4/8 = 0.5$.

**Definition 4 (Data Dependence Similarity — DDS).** *Let X, Y be BPEL processes and* $>X$, $>Y$ *the data dependences,* $>X^{-1}$, $>Y^{-1}$ *the inverse data dependences of their respective BACGs* $(B_X, B_Y)$. *Data Dependence Similarity is defined as*:

$$sim_{>}(B_X, B_Y) = \frac{\left|(>X \cup >X^{-1}) \cap (>Y \cup >Y^{-1})\right|}{\left|(>X \cup >X^{-1}) \cup (>Y \cup >Y^{-1})\right|}.$$

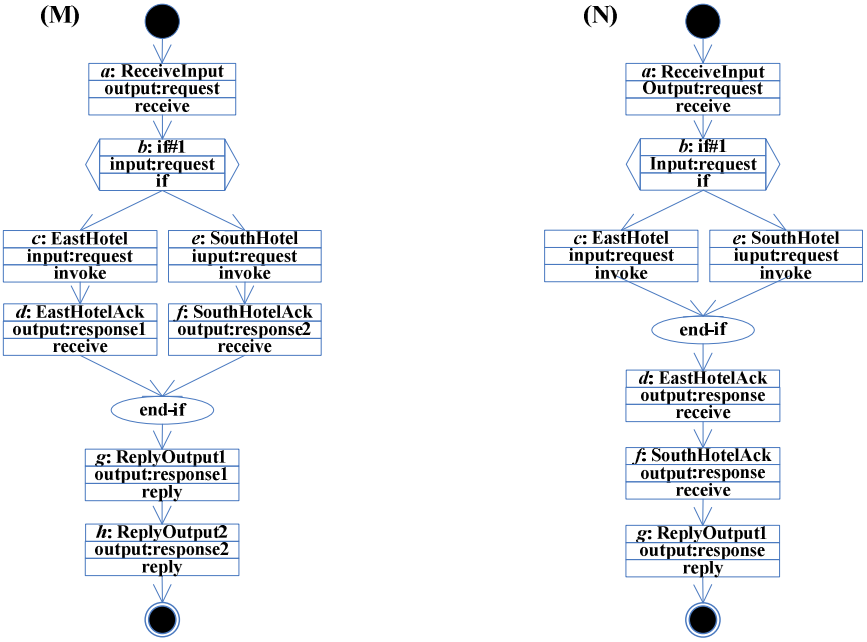



**Fig. 4.** The two variations of BPEL process of the travel agency

The exemplary BPEL processes *M* and *N* show a significant difference in their data dependences, because *h* is missing in *N*, the outputs of *d* and *f* are both the input of *g*, whereas the outputs of *d* and *f* are respectively the inputs of *g* and *h* in *M*. This leads to the following data dependence similarity: $sim_{>}(B_M, B_N) = 8/12 \approx 0.667$.

**Definition 5 (Asyn-invocation Dependence Similarity — ADS).** *Let X, Y be BPEL processes and* $\downarrow X$, $\downarrow Y$ *the asyn-invocation dependences,* $\downarrow X^{-1}$, $\downarrow Y^{-1}$ *the inverse asyn-invocation dependences of their respective BACGs* $(B_X, B_Y)$. *Asyn-invocation Dependence Similarity is defined as*:

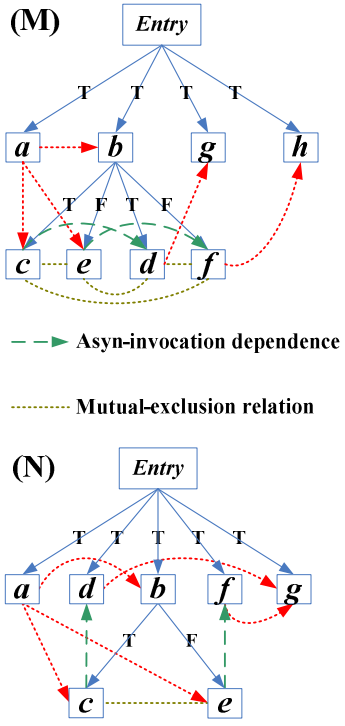
$$sim_{\downarrow}(B_X, B_Y) = \frac{\left|(\downarrow X \cup \downarrow X^{-1}) \cap (\downarrow Y \cup \downarrow Y^{-1})\right|}{\left|(\downarrow X \cup \downarrow X^{-1}) \cup (\downarrow Y \cup \downarrow Y^{-1})\right|}.$$

In Fig.5, we see that there are the asyn-invocation dependences between the pairs (*c*, *d*), (*e*, *f*). The BPEL processes *M* and *N* share the same asyn-invocation dependence relations for (*c*, *d*) and (*e*, *f*). This yields $sim_{\downarrow}(B_M, B_N) = 4/4 = 1$.

The mutual-exclusion is the strictest relation of a BACG, because it enforces the absence of a co-occurrence of two activities within one BPEL process instance. There is a mutual-exclusion relation between any two activities that also contains their inverse pairs from different branches in some structured activities, e.g., <if>, <while>.

**Definition 6 (Mutual-Exclusion Similarity — MS).** *Let X, Y be BPEL processes and +X, +Y the mutual-exclusion relations of their respective BACGs* ($B_X$, $B_Y$). *Mutual-Exclusion Similarity is defined as*:

$$sim_{+}(B_X, B_Y) = \frac{\left|+X \cap +Y\right|}{\left|+X \cup +Y\right|}.$$




| M | *a* | *b* | *c* | *d* | *e* | *f* | *g* | *h* |
|---|---|---|---|---|---|---|---|---|
| *a* | ○ | > | > | ○ | > | ○ | ○ | ○ |
| *b* | < | ○ | → | → | → | → | ○ | ○ |
| *c* | < | ← | ○ | ↓ | + | + | ○ | ○ |
| *d* | ○ | ← | ↑ | ○ | + | + | > | ○ |
| *e* | < | ← | + | + | ○ | ↓ | ○ | ○ |
| *f* | ○ | ← | + | + | ↑ | ○ | ○ | > |
| *g* | ○ | ○ | ○ | < | ○ | ○ | ○ | ○ |
| *h* | ○ | ○ | ○ | ○ | ○ | < | ○ | ○ |

| N | *a* | *b* | *c* | *d* | *e* | *f* | *g* |
|---|---|---|---|---|---|---|---|
| *a* | ○ | > | > | ○ | > | ○ | ○ |
| *b* | < | ○ | → | ○ | → | ○ | ○ |
| *c* | < | ← | ○ | ↓ | + | ○ | ○ |
| *d* | ○ | ○ | ↑ | ○ | ○ | ○ | > |
| *e* | < | ← | + | ○ | ○ | ↓ | ○ |
| *f* | ○ | ○ | ○ | ○ | ↑ | ○ | > |
| *g* | ○ | ○ | ○ | < | ○ | < | ○ |

**Fig. 5.** The BACGs and matrixes of behavioral relations between two activities for *M* and *N*

From the Fig.5, we realize that there are the mutual-exclusion relations between the pairs (*c*, *e*), (*c*, *f*), (*d*, *e*), and (*d*, *f*), as well their inverse pairs respectively in *M*, but only (*c*, *e*) and (*e*, *c*) are mutual-exclusive in *N*. This yields $sim_{+}(B_M, B_N) = 2/8 = 0.25$.

In most cases, the behaviors between activities are reflected by the independence relations of those activities in a BPEL process. The independence similarity strives to reward a large overlap of two independence relations.

**Definition 7 (Independence Similarity — IS).** *Let X, Y be BPEL processes and* $\circ X$, $\circ Y$ *the independence relations of their respective BACGs* ($B_X$, $B_Y$). *Independence Similarity is defined as*:

$$sim_{\circ}(B_X, B_Y) = \frac{|\circ X \cap \circ Y|}{|\circ X \cup \circ Y|}.$$

In the BPEL processes *M* and *N*, there are the independence relations for the other pairs, except that those pairs with the partial order relations or mutual-exclusion relations. This yields $sim_{\circ}(B_M, B_N) = 19/44 \approx 0.432$.

### 3.4 Aggregated Metric for BPEL Activity Constraint Graphs

Based on the elementary similarity measures defined above, we construct an aggregated similarity metric as follows. Each elementary similarity translates into an elementary metric, $d_B(B_P, B_Q) = 1 - sim_B(B_P, B_Q)$ for all $B \in \{\rightarrow, >, \downarrow, +, \circ\}$, as explained in Section 3.3. Then, we sum up these values and assign a weight that accounts for the respective metric's impact on the overall metric. We postulate *B* as the universe of all possible BACGs. In practice this matches the BACGs of all processes within a repository.

**Definition 8 (BPEL Activity Constraint Graph Metric).** $L = (B, d_B)$ *is a metric space of BACGs B, where the BACG metric* $d_B : B \times B \rightarrow \mathbb{R}$ *is a metric, with* $h \in \{\rightarrow, >, \downarrow, +, \circ\}$ *and weighting factors* $w_h \in \mathbb{R}$, $0 < w_h < 1$ *such that* $\sum_h w_h = 1$.

$$d_B(B_X, B_Y) = 1 - \sum\nolimits_h w_h \bullet sim_h(B_X, B_Y).$$

In order to apply to this aggregate metric for similarity search, it has to be proven that it is indeed a metric as established in Definition 1. The process of proof is following:

**Theorem 1.** *The weighted sum* $D(p_m, p_n) = \sum w_h \cdot d_h(p_m, p_n)$ *of elementary metrics* $d_h$ *is a metric if* $\forall h \in [1...n] : w_h \in \mathbb{R} \wedge 0 < w_h < 1$, $\forall p_m, p_n, p_l \in X$, *and* $d_h(p_m, p_n) \in \mathbb{R}$.

*Proof.* $D(p_m, p_n)$ holds the properties symmetry, nonnegativity, identity, and triangle inequality.

- Symmetry: $d_h(p_m, p_n) = d_h(p_n, p_m) \wedge \sum_h w_h = 1 \Rightarrow \sum w_h \cdot d_h(p_m, p_n) = \sum w_h \cdot d_h(p_n, p_m) \Rightarrow D(p_m, p_n) = D(p_n, p_m)$.
- Nonnegativity: $d_h(p_m, p_n) \geq 0 \wedge w_h > 0 \Rightarrow w_h \cdot d_h(p_m, p_n) \geq 0 \Rightarrow \sum w_h \cdot d_h(p_m, p_n) \geq 0 \Rightarrow D(p_m, p_n) \geq 0$.
- Identity: $d_h(p_m, p_m) = 0 \Leftrightarrow D(p_m, p_m) = 0$; $D(p_m, p_n) = 0 \Leftrightarrow \sum w_h \cdot d_h(p_m, p_n) = 0 \wedge w_h > 0 \Leftrightarrow \sum d_h(p_m, p_n) = 0 \Leftrightarrow p_m = p_n$.

- Triangle Inequality: $d_h(p_m, p_n) \leq d_h(p_m, p_l) + d_h(p_n, p_l) \wedge w_h > 0 \Rightarrow w_h \cdot d_h(p_m, p_n) \leq w_h \cdot d_h(p_m, p_l) + w_h \cdot d_h(p_n, p_l) \Rightarrow \sum w_h \cdot d_h(p_m, p_n) \leq \sum w_h \cdot d_h(p_m, p_l) + \sum w_h \cdot d_h(p_n, p_l) \Rightarrow D(p_m, p_n) \leq D(p_m, p_l) + D(p_n, p_l)$.

# 4 Experiments

In this part, we present the experimental evaluation of our similarity measures and the aggregated metric. We describe the setup of our experiments in Section 4.1 and present the results of the experiments in Section 4.2. Finally, the threats of our experiment are analyzed in Section 4.3.

## 4.1 Setup

In order to evaluate the appropriateness of the proposed similarity measures based on BACGs and Jaccard coefficient in practice, our experiments relied on a collection of BPEL processes. The collection comprised 80 processes borrowed and adapted from the Oracle BPEL Process Manager Samples and they were tagged as the "document processes". We then randomly extracted 8 processes from this collection as the "query processes" that have been undergone the modifications. The reason for modifying the query processes was to study the effect of different types of variations in activity constraints on the precision and recall of the proposed metrics.

We performed the experiments by applying the five elementary metrics to perform a similarity search for each of the 8 query processes, i.e., each query process was compared with each of the 80 document processes and the results were ranked from highest to lowest similarity score. Then, we manually determined the relevance for each of the 640 possible pairs. The similarity of each pairs was rated on a 1–7 Likert scale [2, 21]. If a pair received a score of 5 or higher ("somewhat similar" to "very similar"), we considered this pair to be relevant, which means that a query containing the "query process" should return the "document process". Then, the document processes with meeting a score of 5 or higher were considered to be the relevant processes. We had to establish that there was no bias in our relevance judgments, since this rating is only done by ourselves. To this end, we presented a randomly extracted subset of 80 pairs to several BPEL experts from the Asian-Pacific region, and requested them to rate the similarity of each pair on the same Likert scale. Finally, we calculated the inter-rater dependency between our own judgment and those from the experts using the Pearson correlation coefficient. The correlation which was very strong (0.97 Pearson correlation coefficient), showed that our judgement was consistent with the results from those experts. In addition, the similarity of activity labels is quantified based on the string edit distance (SED) [4]. If this similarity degree exceeds a threshold, i.e., 0.8, we assume that two labels are a matching pair.

## 4.2 Evaluation

In this paper, we apply to the relation between recall and precision to evaluate the accuracy and effectiveness of our approach. In the field of information retrieval,

precision measures the ability that the non-relevant information is refused in retrieval system. It is the fraction of retrieved documents that are relevant to the search [22]:

$$\text{precision}=\frac{\left|\{\text{relevant documents}\}\cap\{\text{retrieved documents}\}\right|}{\left|\{\text{retrieved documents}\}\right|}.$$

Precision takes all retrieved documents into account, but it can also be evaluated at a given cut-off rank, considering only the topmost results returned by a system, e.g., for a query on a set of BPEL processes, precision is the number of correct results divided by the number of all returned results. Recall measures the ability that the relevant information is retrieved in a retrieval system. It is in the fraction of the documents that are relevant to the query that are successfully retrieved [22]:

$$\text{recall}=\frac{\left|\{\text{relevant documents}\}\cap\{\text{retrieved documents}\}\right|}{\left|\{\text{relevant documents}\}\right|}.$$

An example is that for a query on a set of BPEL processes, recall is the number of correct results divided by the number of results that should have been returned. It is trivial to achieve recall of 100% by returning all documents in response to any query. Therefore, recall alone is not enough but one needs to measure the number of non-relevant documents also by computing the precision. Actually, there is an antidependence relation between recall and precision, i.e., if we increase the recall value, the precision value will be reduced accordingly, and vice verse.

In the first experiment, we investigated the relation between precision and recall for all metrics. Given one of six recall values in the interval between zero and one, i.e., 0, 0.2, 0.4, 0.6, 0.8, 1.0, we can calculate the corresponding precision of each of five elementary similarities based on a ranked list of the similarity degrees of all pairs above. For each metric, we can obtain 8 corresponding precision values in a recall value since there are 8 query processes in our experiment. Aggregating the results for all queries using the arithmetic mean yields an average precision values for each recall value. Fig.6 (a) describes the precision-recall curve obtained with different metrics. We can find that the independence similarity and data dependence similarity do not achieve the best precision values for all recall levels even though it yields good overall results. This suggests that an aggregated similarity should be applied that combines five elementary metrics in a weighted fashion as introduced in Section 3.4.

In the second experiment, the aggregated metrics based on BACGs is evaluated. Fig.6 (b) shows the result as a precision-recall curve for two metrics. As the baseline metric, behavioral profiles [18] from the field of process models are very similar to our measures and work well. The authors assumed that the weights of all metrics are equal as introduced in [9]. In order to better compare with them, we also applied an aggregated metric that combined all elementary similarities with equal weight based on the results obtained in the first experiment. We can realize that our metrics perform better than the similarity assessment based on behavioral profiles from Fig.6 (b). In particular, the similarity metric based on behavioral profiles declines more obviously than the similarity metric based on BACGs in the interval (0.6, 0.8).

The precision-recall curves also suggest that the differences between both aggregated metrics are rather small for this collection. The relation between precision and recall is an important index to evaluate the performance of the similarity retrieval, and it also can intuitively reflect whether our approach can retrieve the relevant BPEL processes. Based on the relation, we can obtain a compromise value for precision and recall.

As our similarity measures meet the triangle inequality, it is unnecessary to compare each of the query processes with each of all document process in the search. Before retrieving a BPEL repository, we should first do the analysis of cluster for it. Our experiments used the clustering algorithm based on kernel methods [23] to obtain the pivot and a covering radius. According to the clustering information, we can selectively quantify the similarity between each of the query processes and those document processes. This can greatly improve the retrieval efficiency of our approach.

### 4.3 Threats to Validity

As many instances of the BPEL process repository are obtained through manual adjustments, and this repository is somewhat simple in the dimension and structure in our experiments, they tend to be favorable to our metrics. Furthermore, we construct our experiments only containing 80 processes and 8 query processes. The sample is little small. These may lead to the results of our experiments being biased. Then, one major concern is the external validity of our approach, i.e., whether or not our approach is applicable to more other BPEL processes. With this consideration in mind, our approach will be applied to more large-scale BPEL repository in the future.

A limitation of our experiments is that BPEL processes from the collection only cover some basic structures and elements of WS-BPEL 2.0. However, some advanced structures are not fully supported yet, e.g., <for each>, <link>, etc. Thus, our approach is now only applicable to a subset of BPEL processes. This implies a certain threat to the representativeness.

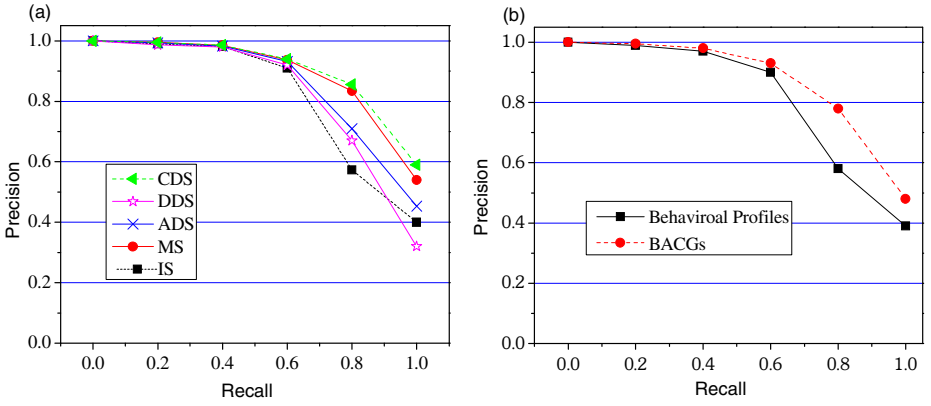



**Fig. 6.** (a) Precision-recall curve for metrics based on five elementary similarities, (b) precision-recall curve for two aggregated metrics based on behavioral profiles and BACGs respectively

## 5 Discussion

In BACGs, only the partial order relations and mutual-exclusion relations can be obtained based on the edges and their labels. In order to quantify the similarity between two BPEL processes from the perspective of the universal set, we introduce an independence relation in BACGs. Actually, it is disputable whether we must introduce the relation. Furthermore, the rigorous definition of the independence relation is missing. This needs future effort of our research group.

Our similarity measure is grounded on the activity constraints (partial order relation, mutual-exclusion relation and independence relation) between in BPEL processes. Actually, these activity constraints have different influence for the similarity between BPEL processes, e.g., if we change control dependence of the corresponding activity of a BPEL process, all corresponding activity pairs with containing this activity may be inconsistent for the two BPEL processes. However, data dependence may only affect the corresponding activity pair. In the situation, we should consider the appropriate weights of different kinds of activity constraints in the definition of five elementary similarity measures. Nevertheless, this is not easy as the weights are somewhat subjective and some domain knowledge also may be considered. Due to calculate the similarity degrees between BPEL processes, this is a boring and heavy job which may manually introduce some unnecessary errors. Thus, we need to implement a proof-of-concept tool that can automatically calculate the similarity degree between BPEL processes in the future work.

The similarity measure is a first attempt to define the similarity of BPEL processes based on BACGs. However, we do not realized how the experts of business processes valuate the similarity of BPEL processes and which kind of similarity measures they perceive to be more appropriate. In the future, through the form of questionnaire, we will present the appropriateness and effectiveness of our approach used an empirical study based on statistical hypothesis testing.

## 6 Related Work

Existing work in the context of determining similarity of BPEL processes can be assigned to three categories: textual similarity, structural similarity, and behavioral similarity (in particular of behavior described in process models) [2, 24].

Text similarity is based on a comparison of the labels that appear in BPEL processes (activity labels etc.), using either syntactic or semantic similarity metrics, or a combination of both. Many approaches have been proposed on determining the similarity degree of labels [4, 25-28], e.g., signature matching [27] and specification matching [28] compare two software components based on descriptions of component, and the string edit distance (SED) [4] is a very appropriate measure in the syntactic level. For the executable business process, the authors proposed that a published process is matched with a required process when the inputs and outputs of the required process match the inputs and outputs of the published process in [29]. In this paper, we adapt some concepts from this aspect for matching activity labels, and we combine these approaches with the calculation of behavioral similarity.

Structural similarity is based on the topology of the processes seen as graphs, possibly taking into account text similarity as well. The graph isomorphism [30, 31] and the graph edit distance (GED) [24, 32] can be commonly used to measure the structural similarity. Unfortunately, these measures usually only examine edges and nodes and cannot catch their syntactical issues. Then, Li et al. use high-level change operations to measure the distance between process models [33]. In [10], the problem of measuring distance between process models is transformed into a graph matching problem. However, two processes may be quite similar in terms of their structure and the labels used, but their behavior may be quite different [6]. Those methods may lead to being low accurate in a certain extent on determining similarity of those processes.

Therefore, the need to take into account the behavior of a process was underlined by some researchers [2, 7-10, 34, 35]. Zha et al. presented a similarity measure based on transition adjacency relations for workflow nets, but the computation of all transition adjacency relations may also be affected by state space explosion [34]. In order to approximate the behavior of process models to assess their similarity, causal footprints have been proposed that capture causal dependencies for activities by sets of causal predecessors and successors for each activity [8]. The proposed similarity measure was based on the notion of so-called "principal transition sequences", which aimed to provide an approximation of the essence of a process model in [7]. The publication [9] introduced a proper metric to quantify process model similarity based on behavioral profiles. Whereas, at present, most of the behavioral similarity does not meet the characteristics of a distance function or time complexity is too high, resulting in a low performance of the behavioral metrics. Furthermore, these measures from the field of process model are not very appropriate in the level of executable BPEL process, since they only regard their control flow but ignore the data information.

Based on activity constraints, we also studied consistency issues between business processes [36-37]. There are three main different from BPEL similarity. First, they do not quantify the scale of the overlap in terms of shared activities of two BPEL processes. Second, in order to compare the set of possible activity pairs of both BPEL processes, all parts that have been subject to projection (in either direction) have to be discarded. Third, the existing measures are no metrics.

## 7 Conclusion and Future Work

In this paper, we introduced a proper metric to quantify behavioral similarity of BPEL processes. This metric is built from five elementary similarity measures that are based on BACGs and the Jaccard coefficient. To show the applicability of our approach in practice, we apply our approach to analyze the similarity between the real-life BPEL processes. Experiments, both with artificial and real-life BPEL processes, were conducted to explore characteristics of the proposed similarity measure.

Our current work still has some limitations. In the future, we will try to address these limitations to improve the applicability and effectiveness of our approach and implement our approach in a prototype tool.

**Acknowledgements.** This work is supported by the National Natural Science Foundation of China (No. 61202003), the Specialized Research Fund for the Doctoral Program of Higher Education (No. 20113219120021), the Major Research Project of Jiangsu Province (No. BK2011022), and the State Key Laboratory of Software Engineering (No. SKLSE2012-09-05).

## References

1. WS-BPEL 2.0 Specification (2007), http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf
2. Dijkman, R., Dumas, M., van Dongen, B., Käärik, R., Mendling, J.: Similarity of business process models: Metrics and evaluation. Inf. Syst. 36(2), 498–516 (2011)
3. Becker, M., Laue, R.: A comparative survey of business process similarity measures. Computer in Industry 63, 148–167 (2012)
4. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics Doklady 10(8), 707–710 (1966)
5. Grigori, D., Corrales, J.C., Bouzeghoub, M.: Ranking BPEL processes for service discovery. IEEE Transactions on Computing Service 3(3), 178–192 (2010)
6. van der Aalst, W., de Medeiros, A.A., Weijters, A.: Process Equivalence: Comparing Two Process Models Based on Observed Behavior. In: Dustdar, S., Fiadeiro, J.L., Sheth, A.P. (eds.) BPM 2006. LNCS, vol. 4102, pp. 129–144. Springer, Heidelberg (2006)
7. Wang, J.M., He, T.F., Wen, L.J., Wu, N.H., ter Hofstede, A.H.M., Su, J.W.: A Behavioral Similarity Measure between Labeled Petri Nets Based on Principal Transition Sequences. In: Meersman, R., Dillon, T.S., Herrero, P. (eds.) OTM 2010. LNCS, vol. 6426, pp. 394–401. Springer, Heidelberg (2010)
8. van Dongen, B., Dijkman, R., Mendling, J.: Measuring Similarity between Business Process Models. In: Bellahsène, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 450–464. Springer, Heidelberg (2008)
9. Kunze, M., Weidlich, M., Weske, M.: Behavioral Similarity — A Proper Metric. In: Rinderle-Ma, S., Toumani, F., Wolf, K. (eds.) BPM 2011. LNCS, vol. 6896, pp. 166–181. Springer, Heidelberg (2011)
10. Corrales, J.C., Grigori, D., Bouzeghoub, M.: BPEL Processes Matchmaking for Service Discovery. In: Meersman, R., Tari, Z. (eds.) OTM 2006. LNCS, vol. 4275, pp. 237–254. Springer, Heidelberg (2006)
11. Lipkus, A.: A Proof of the Triangle Inequality for the Tanimoto Distance. Journal of Mathematical Chemistry 26, 263–265 (1999)
12. Wu, Y.P., Bao, W.D., Zhang, W.M.: Data Matching Method Based on Triangle Inequality Theore. Journal of South China University of Technology 38(7), 33–38 (2010)
13. Chávez, E., Navarro, G., Baeza-Yates, R., Marroquín, J.L.: Searching in Metric Spaces. ACM Comput. Surv. 33(3), 273–321 (2001)
14. Hjaltason, G.R., Samet, H.: Index-driven similarity search in metric spaces (Survey Article). ACM Trans. Database Syst. 28(4), 517–580 (2003)
15. Ferrante, J., Ottenstein, K.J., Warren, J.D.: The program dependence graph and its use in optimization. TOPLAS 9(3), 319–349 (1987)
16. Song, W., Ma, X.X., Cheung, S.C., Hu, H., Yang, Q.L., Lü, J.: Refactoring and publishing WS-BPEL processes to obtain more partners. In: ICWS 2011, pp. 129–136 (2011)
17. Song, W., Tang, J.H., Zhang, G.X., Ma, X.X.: Substitutability analysis of WS-BPEL sevices. China Science: Information Science 42(3), 264–279 (2012) (in Chinese)

18. Weidlich, M., Mendling, J., Weske, M.: Efficient Consistency Measurement based on Behavioural Profiles of Process Models. IEEE Trans. Softw. Eng. (2011)
19. Nanda, M.G., Chandra, S., Sarkar, V.: Decentralizing execution of composite Web services. In: OOPSLA 2004, vol. 39(10), pp. 170–187 (2004)
20. Kŏnig, D., Lohmsnn, N., Moser, S.: Extending the compatibility notion for abstract WS-BPEL processes. In: WWW 2008, pp. 785–794 (2008)
21. Likert scale, https://en.wikipedia.org/wiki/Likert_scale
22. Precision and Recall, http://en.wikipedia.org/wiki/Precision_and_recall
23. Cooper, K.D., Harvey, T.J., Kennedy, K.: Iterative Data-flow Analysis. ACM (2002) (revisited)
24. Dijkman, R., Dumas, M., García-Bañuelos, L.: Graph Matching Algorithms for Business Process Model Similarity Search. In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) BPM 2009. LNCS, vol. 5701, pp. 48–63. Springer, Heidelberg (2009)
25. Euzenat, J., Shvaiko, P.: Ontology matching. Springer, Heidelberg (2007)
26. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. VLDB J. 10(4), 334–350 (2001)
27. Zaremski, A.M., Wing, J.M.: Signature matching: a tool for using software libraries. ACM TOSEM 4(2), 146–170 (1995)
28. Zaremski, A.M., Wing, J.M.: Specification matching of software components. ACM TOSEM 6(4), 333–369 (1997)
29. Paolucci, M., Kawamura, T., Payne, T.R., Sycara, K.: Semantic matching of Web services capabilities. In: Horrocks, I., Hendler, J. (eds.) ISWC 2002. LNCS, vol. 2342, pp. 333–347. Springer, Heidelberg (2002)
30. Babai, L., Erdoŝ, P., Selkow, S.M.: Random graph isomorphism. SIAM J. Comput. 9(3), 628–635 (1980)
31. Krissinel, E.B., Henrick, K.: Common subgraph isomorphism detection by back-tracking search. Softw. Pract. Exper. 34(6), 591–607 (2004)
32. Bunke, H.: On a relation between graph edit distance and maximum common subgraph. Pattern Recognition Letters 18(8), 689–694 (1997)
33. Li, C., Reichert, M., Wombacher, A.: On Measuring Process Model Similarity Based on High-Level Change Operations. In: Li, Q., Spaccapietra, S., Yu, E., Olivé, A. (eds.) ER 2008. LNCS, vol. 5231, pp. 248–264. Springer, Heidelberg (2008)
34. Zha, H., Wang, J., Wen, L., Wang, C., Sun, J.: A workflow net similarity measure based on transition adjacency relations. Computers in Industry 61, 463–471 (2010)
35. Huang, Z.C., Huai, J.P., Liu, X.D., Li, X., Zhu, J.J.: Automatic Service Discovery Framework Based on Business Process Similarity. J. of Sof. 23(3), 489–503 (2012)
36. Zhang, X.W., Song, W., Xing, J.C., Yang, Q.L., Wang, H.D., Zhang, W.J.: Behavioral Consistency Measurement and Analysis of WS-BPEL Processes. In: Wang, J., Xiong, H., Ishikawa, Y., Xu, J., Zhou, J. (eds.) WAIM 2013. LNCS, vol. 7923, pp. 619–630. Springer, Heidelberg (2013)
37. Song, W., Zhang, W.J., Zhang, G.X., Ding, J.H., Zhang, X.W.: Quantifying Consistency between Conceptual and Executable Business Processes. In: SCC 2013, pp. 9–16 (2013)

# Process Model Storage Solutions: Proposition and Evaluation

Jie Li[1], Lijie Wen[1], Jianmin Wang[1], and Zhiqiang Yan[1,2]

[1] School of Software, Tsinghua University, Beijing 100084, P.R. China
[2] School of Information, Capital University of Economics and Business, 100070, P.R. China
nju_carol@hotmail.com, wenlj00@mails.thu.edu.cn,
jimwang@tsinghua.edu.cn, zhiqiang.yan.1983@gmail.com

**Abstract.** With the development of Business Process Management (BPM) technology, more and more organizations use process models to describe their business processes. These process models are modified frequently for higher management efficiency and effectiveness. This causes each process models have multiple versions. Since those models focus on the same business behavior, they are similar in structure. Based on process model similarity, model storage can be improved. To store these models efficiently, five process model storage solutions are presented. Experiments are designed to compare those solutions from time and space aspects.

**Keywords:** Petri Nets, process model, similarity measure, model storage.

## 1 Introduction

With fast development of BPM technology, it is common for large enterprises to maintain thousands of process models. To handle variable needs of clients and improve the quality of business processes continuously, these process models change frequently. Therefore, different versions of the same process model appear. Those model versions help decision makers analyze the change rules of business process, which provide a guideline for future change and improvement. They are important treasures of the organizations.

Generally, different versions of the same model describe the same business behavior, which makes them similar in structure. A process model may be changed by different stakeholders. By saving the edit operations happened during the change process, users can easily trace the change history. Meanwhile, it can also reduce the storage space taken by process models. This is useful in some situations. For example, when the cloud workflow engine is working, thousands of process models are stored in its memory. Compress the model storage space can reduce memory usage and help the engine serve more clients at the same time. This idea is similar with SVN, which shows a great performance in the version control of source codes. In process repositories, version management needs to be specialized in order to meet process specific requirements [1]. However, as far as we know, no mature solution of this idea is presented to manage the storage of process models so far.

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-319-02922-1_10

M. Song, M.T. Wynn, and J. Liu (Eds.): AP-BPM 2013, LNBIP 159, pp. 56–66, 2013.

Based on above requirements, we come up with five process model storage solutions. Experiments are run with both industrial and artificial process model collections to evaluate these storage solutions.

The remainder of this paper is organized as follows. Section 2 proposes five model storage solutions. After that, experiment results and analysis are presented in Section 3. Section 4 summarizes related work. We conclude with a summary and outlook in Section 5.

## 2 Storage Solutions of Process Model

Nowadays, the amount of business process models rapidly grows and process models of the same business behavior are similar in structure. Based on this reality, five different process model storage solutions are proposed in this section.

The first storage solution, named Direct Storage Solution (DSS) simply keeps every model version on the disk, while the Log Storage Solution (LSS) only saves the initial model and change logs, which records all edit operations happened in the changing process. Interval Storage Solution (ISS) makes a compromise between the former two solutions. It stores model version *ki* where *i* represents the specified interval and *k* is zero or a positive integer. Based on the concept of reference model in model variants management, the Nearest Model Storage Solution (NMSS) saves the nearest model version, which has the highest similarity with all other versions. Finally, the fifth solution-Interval Nearest Model Storage Solution (INMSS) is the combination of ISS and NMSS.

Suppose the initial model version is 0 and the newest model version is n. Five process model storage solutions are presented in details below.

### 2.1 Direct Storage Solution (DSS)

The easiest way to save process models is keeping every version of the model. That is, directly storing all model version files (0, 1, 2, …, n) on the disk. The biggest advantage of DSS is the shortest response time. When a user asks for one model version, the model file can be directly returned, without any additional computing time. However, with continuous change of the model, storing all versions of model will take more and more disk space, which is a big challenge for model management.

### 2.2 Log Storage Solution (LSS)

Similar storage problem occurred in the storage of programming source code. Version control system perfectly solved this problem. The core idea of version control system is change management, which can be brought to process model storage. If all the edit operations happened in the changing process are known beforehand, we can easily derive any model version from the initial model and that is the LSS: storing the initial model version file and a change log. Every change of the model can be recorded in the change log. When a user request comes, change log will be loaded and the

recorded edit operations will be performed on the initial model until we get the target version. Compared with DSS, LSS only keeps one model file and one log file on the disk, which takes little disk space and makes it easy for long-term backup. While LSS shows a great performance on space, it may meet with some problems with response time. When the version number grows up, it will take more edit operations to get the newest model version, which means more additional computing time and longer response time. That may lead to bad user experience in the end.

### 2.3 Interval Storage Solution (ISS)

From the above, we know that DSS needs large disk spaces and LSS costs much response time. ISS makes a compromise between those two solutions: storing several model versions and a change log. Specifically, let the version interval be $i$, then model version $ki$, $k = 0,1,2,3, \ldots$ will be kept on the disk. Meanwhile, the change log records the edit operations needed for transforming model $ki$ to model $ki + 1, ki + 2, \ldots, k(i + 1) - 1$. Suppose the user needs to get model version $x$, if $x = ki$, directly return the model file, otherwise find the maximum value of $k$ which satisfies $ki < x$, read the change log and perform edit operations on model $ki$ until we get model $x$. In this way, it takes at most $i - 1$ times of changes to get model $x$, which provides an acceptable response time. Value of interval $i$ is important for ISS, we can get a good balance between time and space by adjusting the value of $i$. If $i = 1$, ISS is the same with DSS; if $i > n$, ISS is the same with LSS. In real world applications, we should confirm the cost of time and space at first, and then find the best value of $i$, which makes the total cost minimal.

### 2.4 Nearest Model Storage Solution (NMSS)

In process-aware information system, there are a large number of process model variants which are derived from the same process model, but slightly differ in structure. As a response to this situation, Li et al. come up with a solution. That is, discovering a reference model out of which the variants can be configured with minimum efforts [2]. This idea can be used to improve LSS. Now not only the initial model and change log are saved on the disk, but also a nearest model. Here nearest model is the model version which has the minimum distance to all other model versions. The distance can be measured by the process model similarity metric introduced in [3], which applies graph edit distance to process model and gives a clear description of the difficulty for transforming one model to another. Every time when the version grows to a threshold, the nearest model will be recomputed and saved on the disk. Suppose the nearest model version is $r$ and the user wants to get model $x$. If $x = r$, directly return model $r$. If $x > r$, then read the change log, perform $x - r$ times of changes on model $r$ until we get model $x$. Otherwise if $x < r$, perform $x$ times of change on model 0. The NMSS performs at most $\max\{x - r, x\}$ times of change. Therefore it takes less additional computing time than LSS. Meanwhile, it only takes a little more spaces to store the nearest model.

### 2.5 Interval Nearest Model Storage Solution (INMSS)

In NMSS, the response time can be reduced only when the model version is bigger than nearest model version. To solve this problem, we bring forward INMSS by combining ISS and NMSS. That is, let the interval be i, compute a nearest model for every i model versions and save them on the disk together with the change log. To make it clearer, suppose in model set $\{ki, ki+1, ki+2, \ldots, (k+1)i-1\}$ and $k = 0,1,2,3, \ldots$, $r_k$ is the nearest model to other models, then model $r_0, r_1, r_2, \ldots$ will be stored on the disk. By adding background computation work, INMSS takes nearly the same disk space with ISS, but it can further reduce the response time. Compared with NMSS, INMSS cuts down the maximum times of change in every user request to $i-1$ by taking more disk space. Moreover, consider the background computation, when the model version number increase to some degree, NMSS has to compute the nearest model for all model versions again, which costs a lot of time. However, INMSS only needs to get the nearest model for i new models.

Above all, we introduce five different process model storage solutions. Every solution has its advantages and disadvantages. In real world applications, both time and space requirements have to be carefully considered before we determine which solution to be used. In fact, it's also a good idea to use two or more solutions together. For example, when we start to save the process models, use DSS to store all model versions on disk. After the amount of models increases to a threshold, we transform to ISS. What we need to do is to compute the edit operations between models using process model similarity metrics [3], write the edit operations into change log and then delete models $ki+1, ki+2, \ldots, k(i+1)-1$, $k = 0,1,2,3, \ldots$ from the disk to save spaces. In next section, we will analyze and evaluate the five storage solutions by experiments.

## 3 Experiments

In this section, we use experiments to analyze and compare the five storage solutions on both industrial and artificial model collections. Experiment results are evaluated from the viewpoint of response time and disk spaces. Specifically, for every model data set, we use each storage solution to obtain all models in the set. Meanwhile, the total response time and disk space taken by the storage solution are recorded.

During the experiment process, we used Greedy Algorithm [3] to measure the similarity between process models. The process models are all represented by Petri net. We experimented with different values of the parameters and found the optimal setting among those we tested is: wskipn=1, wskipe=1, wsubn=3 and ledcutoff=0.5 (these parameters are presented in [3], where wskipn represents the weight for skipped nodes; wskipe represents the weight for skipped edges; wsubn represents the weight of substituted nodes). Every experiment process is repeated 5 times and average values are presented.

All experiments are completed on one notebook computer. This computer uses Intel(R) Core(TM) i3 CPU M350 @ 2.27GHz and 2.00GB memory. The experiment program is implemented in JAVA. We run it on 64-bit Windows 7 system with JDK 1.6.0, and the memory configuration of JVM is 512MB.

### 3.1 Industrial Data Set

First we make an evaluation of the storage solutions on industrial data sets. The data sets consist of 102 process models from Dongfang Boiler Group Corporation. Those process models belong to five business modules: cost, finance, material, production and sale. Process models from the same business module are similar in structure. According to their business module, we divided them into five data sets, as shown in Table 1.

**Table 1.** Basic information of industrial data sets

| Business module | Model number | Avg. transitions | Avg. places | Avg. arcs |
|---|---|---|---|---|
| cost (C) | 18 | 8.78 | 8.94 | 18.72 |
| finance (F) | 29 | 8.38 | 7.76 | 16.66 |
| material (M) | 37 | 11.08 | 10.30 | 21.51 |
| production (P) | 12 | 9.42 | 8.83 | 19.08 |
| sale (S) | 6 | 7.83 | 6.83 | 14.33 |

We apply five storage solutions to the industrial data sets. In ISS and INMSS, the interval value is set to 3. The experiment results are shown in Fig. 1.

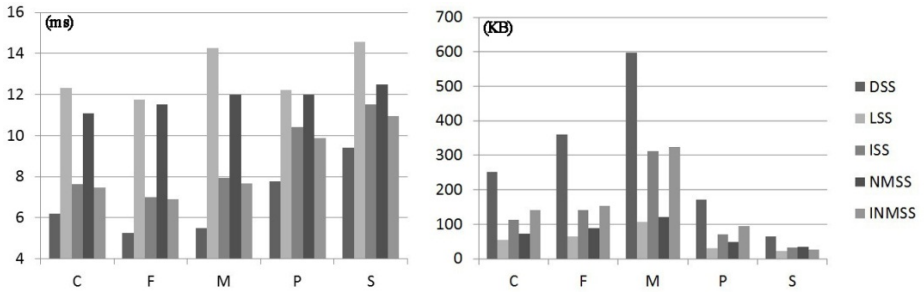



**Fig. 1.** Comparison of storage solutions on industrial data sets from time and space aspects

The left graph of Fig. 1 compares the storage solutions from the time aspect. The horizontal axis represents five business modules, while the vertical axis represents the average response time of the storage solution on each business module and the unit is millisecond. The average response time for one version is obtained by dividing the total response time by the number of model versions. From the graph, we can see that LSS takes the highest response time on every data set, while DSS takes the lowest. ISS, NMSS and INMSS are in the middle.

The right graph of Fig. 1 compares the storage solutions from the space aspect. The horizontal axis represents five business modules and the vertical axis represents the disk space taken by the storage solution. The unit of disk spaces is kilobyte. As we can see, DSS takes the largest disk space on every data set. Disk space taken by ISS and INMSS is much smaller, while LSS and NMSS take the smallest spaces.

### 3.2 Artificial Data Set

We also run the experiment on artificial data sets. In order to simulate real storage situation of process models, we use the model generation function of BeehiveZ platform [4] to generate 3 initial Petri net models, which are different in size. Their basic information is given in Table 2.

**Table 2.** Basic information of initial models

| Model number | Transition number | Place number | Arc number | Model size (byte) |
|---|---|---|---|---|
| 1 | 46 | 14 | 92 | 66,819 |
| 2 | 65 | 32 | 142 | 101,908 |
| 3 | 104 | 46 | 216 | 157,634 |

When the size of model is small, the model loading time will be tiny. This may bring some deviations to the experiment statistics. Therefore we select 3 relatively large process models, as the initial models.

In order to get different model versions based on the initial models, we develop a randomly model editing program, which accepts a model as input and randomly performs edit operations on the model. In order to keep every model sound, we follow the reduction rules [5] to define the random edit operations. The new model version generated by the program will be stored and the edit operation will be written into the change log. Note that in the experiment, only the total number of edit operations is specified, more detailed settings are all randomly determined, for example, the number of each edit operation type and the exact edited object (transition, place or arc). After running randomly editing program on each initial model, 5 different datasets are generated, as shown in Table 3.

**Table 3.** Model storage datasets

| Dataset number | Initial model number | Transition edit operations | Place edit operations | Model number |
|---|---|---|---|---|
| 1 | 1 | 7 | 3 | 10 |
| 2 | 1 | 21 | 29 | 50 |
| 3 | 1 | 60 | 40 | 100 |
| 4 | 2 | 278 | 222 | 500 |
| 5 | 3 | 565 | 435 | 1000 |

In Table 3, the transition/place edit operation column shows the total number of transition/place insertion, transition/place deletion and transition/place substitution operations happened in the editing process. About the random edit process, there are two things need to be noted. First, when a transition/place is added or deleted, the arcs directly connecting to it are added or deleted as well. Therefore we do not define arc insertion/deletion operation separately in the random editing process, but arc edit

operations actually happen together with the transition/place edit operations. Second, every transition/place label is unique before and after the substitution operation.

Table 4, Table 5 and Fig. 2 show the experiment results.

**Table 4.** Average response time for one model on model storage data sets (unit: ms)

| Storage solution \ Dataset number | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| DSS | 12 | 10 | 13 | 32 | 107 |
| LSS | 14 | 13 | 17 | 79 | 293 |
| ISS | 14 | 11 | 15 | 43 | 120 |
| NMSS | 14 | 12 | 16 | 60 | 210 |
| INMSS | 14 | 11 | 15 | 42 | 119 |

**Table 5.** Disk space taken by storage solutions on model storage data sets (unit: byte)

| Storage solution \ Dataset number | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| DSS | 671,923 | 3,344,437 | 8,966,260 | 86,453,660 | 353,334,297 |
| LSS | 68,521 | 77,437 | 89,244 | 211,168 | 391,280 |
| ISS | 249,450 | 726,457 | 1,880,262 | 8,929,807 | 35,898,405 |
| NMSS | 123,965 | 143,827 | 191,019 | 416,234 | 815,046 |
| INMSS | 307,156 | 793,108 | 1,970,492 | 9,138,662 | 36,335,215 |

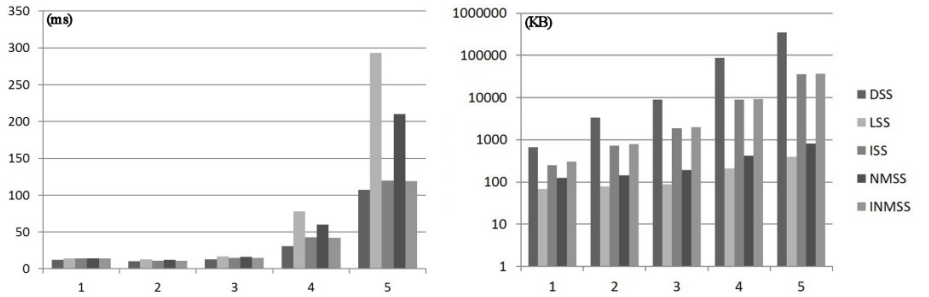



**Fig. 2** Comparison of storage solutions on artificial data sets from time and space aspects

Let's first look at result of DSS, which stores the full process model versions. From the viewpoint of response time, since dataset 1, 2 and 3 are based on the same initial model, after random editing, their model versions are similar in size. Meanwhile, response time of DSS is mainly determined by the model loading time. Therefore we can see from Table 4 that the average response time is nearly the same for dataset 1, 2 and 3. Compared with dataset 1, 2 and 3, the initial model of dataset 4 and 5 are larger, which makes the average response time longer. Combine Table 2 and Table 5, we can find that the disk space taken by DSS nearly shows a linear growth with the number of model versions.

LSS only stores the initial model and change log. For every model version, the change log records the basic edit operations needed for changing this version to the next version.

Observe Table 2 and Table 5, we can see that disk space of LSS is mainly determined by the size of initial model. Space taken by the change log is small and grows slowly. However, average response time grows very fast with the increase of model versions. One reason is that the growth of change log leads to more log loading time. Another reason is the longer additional computing time. In order to get a higher model version, more edit operations need to be performed. When there are a huge number of model versions or multiple user requests in parallel, the response time may become unacceptable.

In Table 5, for dataset 1, LSS takes one tenth disk space of DSS. Also for dataset 5, DSS takes 900 times more space than LSS. Meanwhile, for dataset 5, the average response time of LSS is less than three times of DSS. This means LSS saves the disk space effectively when the model version grows. If the number of model versions is not huge or the requirement of response time is not high, LSS is a good choice.

In ISS, the model version ki, $k = 0,1,2,3, \ldots$ and change log are stored. Here the change log is a little different with the change log in LSS. Now the change log has no need to record the edit operations for growing from model version $ki - 1$ to ki. In Table 4 and Table 5, the interval value is 3 for data set 1, 5 for data set 2 and 3, 10 for data set 4 and 5.

From Table 5, we can see that the disk space taken by DSS takes is i times as much as ISS. At the same time, the response time of ISS doesn't get much longer. ISS obtains fast response at the expense of some disk space. It shows a good balance between time and space.

The nearest model has to be computed before we run NMSS. We get the nearest model of every dataset using the similarity measure algorithm stated in Section 2. NMSS stores the initial model, nearest model and change log. The size of initial model and nearest model determines the total disk space. Therefore it doesn't take much more space than LSS. However, compared with LSS, we can see that NMSS significantly reduce the average response time using the nearest model on dataset 4 and 5.

INMSS keeps the nearest models for every i model versions and change log on disk. Like NMSS, the nearest models need to be computed by similarity measure algorithm. Value of i is the same with the settings in ISS. From Table 5, we can see that ISS and INMSS take nearly the same disk space. INMSS slightly reduces the response time, but not much.

Above all, we introduce the experiments of all storage solutions. From the time aspect, as shown in the left graph of Fig. 2, LSS has the fastest growth of response time, NMSS is in the middle, DSS, ISS and INMSS are similar and they are all slow in growth. We all know that the response time taken by DSS is mainly determined by the model size. Therefore from the viewpoint of time, DSS, ISS and INMSS are the best choices. The result is similar with former experiment on artificial data sets.

From the space aspect, as we can see in the right graph of Fig. 2 (Note that we make a logarithm based on 10 for the vertical axis in order to make it clearer), with the increase of model version number, disk space taken by DSS shows the fastest

growth. Growth rates of ISS and INMSS are nearly the same but slower than DSS. We find LSS and NMSS are the slowest in disk space growth. Therefore from the viewpoint of space, LSS and NMSS are the best choices. The result is also similar with former experiment on artificial data sets.

To summarize the experiment results, take time and space together into consideration, among five storage solutions, ISS and INMSS show better performance. Since INMSS needs additional computation for nearest models, ISS should be the best one. In real world application, we should carefully choose the storage solution by computing the cost of time and space.

## 4 Related Work

To the best of our knowledge, two groups of researchers worked on version management of process models [6,7]. Ekanayake et al. propose a version control method in the process repository based on fragment [6]. Since process models in repository have duplicate fragments, by saving the process fragments instead of full process model versions, the repository size can be reduced. Our work is also based on the structural similarity between process models, but we store the change operations, which makes the space usage even smaller and makes it easier to check the change history. Zhao et al. propose a version preserving directed graph (VPG) [8], which is rather complex, since a VPG maintains information of all versions for a process models. It does not mention the storage problem also does not provide an evaluation with process model collections.

Other than that, Li et al. propose that a reference model can be found by mining algorithm [2]. Based on the reference model, user can get all model variations with minimum efforts. NMSS and INMSS also need to find the nearest model. However, Li et al. use the minimum number of edit operations to measure the distance between process models while we compute the similarity for different edit operations separately. Also, Li et al. focus on process model mining, while we pay attention to the storage of process model.

Another related topic is process similarity search. Many activities in business process management need to compute the similarity between two process models, for example process retrieval [9,10], process mining, process discovery and process integration [11]. Currently, the similarity of process models can be measured on the basis of three aspects: label [12], structure [7,12,13] and behavior [14,15]. The edit distance algorithm stated in this paper is one of the similarity measures based on structure. As an important method for measuring graph similarity, graph edit distance is widely applied in pattern recognition and computer vision field. There are many related research on this problem [16]. Dijkman applies graph edit distance to the similarity search of business process model and comes up with four heuristic algorithms [3]. In this paper, we use one of the algorithms: greedy algorithm to measure the structural similarity between Petri nets.

## 5 Summary and Outlook

Nowadays, business process models are continually evolving. Based on this reality, this paper comes up with five model storage solutions and uses experiments to make comparisons. Advantages and disadvantages of those storage solutions are given, which is valuable for related research in business process model management.

In this paper, we use greedy algorithm to compute the edit distance, which means we may not get the minimum distance between models. The balance between precision and time complexity should be further explored in the future work. At the same time, we are considering to integrate ISS to current open-source code version control system and extend it to real applications.

**Acknowledgements.** This work is supported by the National Natural Science Foundation of China (No. 61003099), the National High-Tech. R&D Program of China (No. 2012AA040904), and the Ministry of Education and China Mobile Research Foundation (No. MCM20123011).

## References

1. Yan, Z., Dijkman, R., Grefen, P.: Business process model repositories–Framework and survey. Information and Software Technology 54(4), 380–395 (2012)
2. Li, C., Reichert, M., Wombacher, A.: Mining Business Process Variants: Challenges, Scenarios, Algorithms. Data & Knowledge Engineering 70(5), 409–434 (2011)
3. Dijkman, R.M., Dumas, M., García-bañuelos, L.: Graph Matching Algorithms for Business Process Model Similarity Search. In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) BPM 2009. LNCS, vol. 5701, pp. 48–63. Springer, Heidelberg (2009)
4. Wu, N.H., Jin, T., Zha, H.P., He, T.F., Wen, L.J., Wang, J.M.: BeehiveZ: An Open Framework for Business Process Model Management. Journal of Computer Research and Development 47(z1) (2010)
5. Murata, T.: Petri nets: Properties, analysis and applications. Proceedings of the IEEE 77(4), 541–580 (1989)
6. Ekanayake, C.C., La Rosa, M., Ter Hofstede, A.H., Fauvet, M.C.: Fragment-based version management for repositories of business process models. In: Meersman, R., Dillon, T., Herrero, P., Kumar, A., Reichert, M., Qing, L., Ooi, B.-C., Damiani, E., Schmidt, D.C., White, J., Hauswirth, M., Hitzler, P., Mohania, M. (eds.) OTM 2011, Part I. LNCS, vol. 7044, pp. 20–37. Springer, Heidelberg (2011)
7. Yan, Z., Dijkman, R., Grefen, P.: Fast Business Process Similarity Search. Distributed and Parallel Databases 30(2), 105–144 (2012)
8. Zhao, X., Liu, C.: Version management in the business process change context. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 198–213. Springer, Heidelberg (2007)
9. Jin, T., Wang, J., Wen, L.: Efficient retrieval of similar workflow models based on structure. In: Meersman, R., Dillon, T., Herrero, P., Kumar, A., Reichert, M., Qing, L., Ooi, B.-C., Damiani, E., Schmidt, D.C., White, J., Hauswirth, M., Hitzler, P., Mohania, M. (eds.) OTM 2011, Part I. LNCS, vol. 7044, pp. 56–63. Springer, Heidelberg (2011)

10. Jin, T., Wang, J., Wen, L.: Efficient retrieval of similar workflow models based on behavior. In: Sheng, Q.Z., Wang, G., Jensen, C.S., Xu, G. (eds.) APWeb 2012. LNCS, vol. 7235, pp. 677–684. Springer, Heidelberg (2012)
11. Bae, J., Liu, L., Caverlee, J., Zhang, L.J., Bae, H.: Development of distance measures for process mining, discovery and integration. Int. J. Web Service Res. 4(4), 1–17 (2007)
12. Dijkman, R., Dumas, M., García-Banñuelos, L., Käärik, R.: Aligning business process models. In: Proc. of EDOC 2009, Auckland, New Zealand (September 2009)
13. Messmer, B.: Efficient Graph Matching Algorithms for Preprocessed Model Graphs. PhD thesis, University of Bern, Switzerland (1995)
14. Zha, H., Wang, J.M., Wen, L.J., Wang, C.K., Sun, J.G.: A workflow net similarity measure based on transition adjacency relations. Computers in Industry 61(5) (2010)
15. Wang, J.M., He, T.F., Wen, L.J., Wu, N.H., Ter Hofstede, A.H., Su, J.: A behavioral similarity measure between labeled petri nets based on principal transition sequences. In: Meersman, R., Dillon, T.S., Herrero, P. (eds.) OTM 2010. LNCS, vol. 6426, pp. 394–401. Springer, Heidelberg (2010)
16. Gao, X.B., Xiao, B., Tao, D.C., Li, X.L.: A survey of graph edit distance. Pattern Anal. Appl. (PAA) 13(1), 113–129 (2010)

# Clustering and Operation Analysis for Assembly Blocks Using Process Mining in Shipbuilding Industry

Dongha Lee[1], Jaehun Park[2], Iq Reviessay Pulshashi[2], and Hyerim Bae[2,*]

[1] Central R&D Institute,
Daewoo Shipbuilding & Marine Engineering Co., Ltd.,
1, Aju-dong, Geoje-si, Gyeongsangnam-do 656-714, Korea
dongha@dsme.co.kr
[2] Department of Industrial Engineering,
Pusan National University,
Busandaehak-ro 63 Beon-gil, Geumjeong, Busan 609-735, Korea
{pjh3479,pulshashi,hrbae}@pusan.ac.kr

**Abstract.** A block assembly process in the shipbuilding industry consists of many work stages. Block assembly involves many workers in many shops. Each assembly block, which is a part of a ship, has a different structure requiring specific work processes. Therefore, in order to better understand such real processes, an information system for monitoring of block position has been developed. Recently, the necessity of using data accumulated in information systems has become greater. This paper proposes a new, clustering and operation analysis method for assembly blocks based on process mining techniques suitable for the shipbuilding industry. The approach consists of four steps: 1) trace clustering from the task perspective, 2) trace clustering from the work shop perspective, 3) definition of new clusters considering task and work shop simultaneously, and 4) comparison of new clusters with other clusters from the process perspective. The output of clustering and operation analysis can be used for production planning purposes such as resource allocation and operation scheduling for assembly blocks. The effectiveness of the proposed method was verified in a case study using real event logs generated from the Block Assembly Monitoring System (BAMS), an information system.

**Keywords:** Shipbuilding, block assembly, process mining, trace clustering, process model.

## 1 Introduction

Generally, a shipbuilding process consists of many operations including block division, cutting/bending, block assembly, outfitting, painting, and erection. Block division, the initial operation in the shipbuilding process, entails division of the planned ship into hundreds of properly sized blocks. Each block is assembled in an assembly shop, after which it undergoes outfitting and painting operations. In the final

* Corresponding author.

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-319-02922-1_10

M. Song, M.T. Wynn, and J. Liu (Eds.): AP-BPM 2013, LNBIP 159, pp. 67–80, 2013.

operation, the structure is erected as a ship in a pre-erection area or on the main dock. Each assembly block, having a unique structure and requiring different work processes, consists of many work stages and involves many workers. Not surprisingly, understanding and managing all of the particular aspects of real shipyard processes is a great challenge. In fact, the production scheduler's resource allocation and operation scheduling are the top priorities of the shipbuilding industry.

This paper focuses on understanding the actual operations of block assembly using log data generated by a monitoring system. By clustering blocks based on the characteristics of the operations those blocks undergo, we can better understand block assembly operations and, on that basis, formulate better production planning guidelines. Per cluster, a production scheduler can more easily and effectively make plans for resource allocation and scheduling. This paper proposes a new, clustering and operation analysis method for assembly blocks using process mining techniques. This paper is organized as follows. Section 2 provides a brief overview of process mining and the related research. Section 3 introduces the new approach for clustering and operation analysis of assembly blocks. Section 4 discusses a case study according to which all of the important principles of the new method are effectively explained. Section 5 summarizes our work and draws conclusions.

## 2 Process Mining and Related Research

Process mining is a technique for extracting process models from execution logs. It aims at improving business processes by providing techniques and tools for discovering processes and managing data as well as organizational and social structures from event logs; the basic idea of process mining, in other words, is to diagnose business processes by mining event logs for knowledge.

Process mining has three different perspectives: (1) the process perspective, (2) the organizational perspective, and (3) the case perspective. The process perspective focuses on the control-flow, that is, the precedence relations of activities. The goal of mining from this perspective is to find a good characterization of all possible paths. The organizational perspective focuses on the originator field, or in other words, on which performers are involved and how they are related. The goal is to either structure the organization by classifying people in terms of roles and organizational units or to show the relations between individual performers. The case perspective focuses on the properties of cases. A case can be characterized by its path in the process or by the originators working on it. A case also can be characterized by the values of the corresponding data elements [1]. In this paper, these three different perspectives are considered in our step-by-step analysis of the characteristics of block assembly operations.

The main emphasis of process mining research has been the extraction of process models only from event logs, which approach has been applied mainly in the

healthcare and service fields. Recently, the application has been extended to other fields [2,6]. In the shipbuilding industry, several attempts have been made to find the processes of block movement. Lee et al.[3] used a data mining technique to cluster blocks, and then analyzed process models independently by group. Lee and Bae[4] derived an analysis framework that begins with the understanding of the entire process and ends with the detection of exceptional processes, both of which methods are based on process mining techniques. However, as far as we know, there has been no study on finding the processes of block assembly operations. This paper deals with clustering and operation analysis of assembly blocks, the issues of which differ from those of block movement. Assembly blocks, which represent parts of a ship, all have unique structures and operations, and so production schedulers, in consideration of resource constraints, have to plan separately for each block. This is to say that in order to maximize resource utilization, schedulers allocate resources and determine working times according to the specific characteristics of respective assembly blocks. Production schedulers keep the groups of assembly blocks to give same processing time and to allocate work shop. In this paper, we mention the group as plan cluster in following sections. Plan clusters have been determined, case by case, from experience. This paper focuses on the clustering of assembly blocks and the analysis of those operations based on the actual data from a monitoring system.

## 3 Proposed Approach

The clustering and operation analysis method proposed in this paper consists of the four steps shown in Fig. 1. The first step is trace clustering from the task perspective. Here, the task is the characteristics of work types to be assembled, and the output is a cluster of assembly blocks that share similar work types. This is called the Task Cluster (TC). The second step is trace clustering from the work shop perspective. Here, a cluster of assembly blocks sharing similar work shops, the Shop Cluster (SC), is obtained. The third step is the definition of new clusters using TC/SC matrix. As the clusters are determined for the same analysis dataset two times respectively, all assembly blocks are located in the TC/SC matrix. Cells intersected by TC and SC are defined by new clusters (NC) when there are blocks in the intersected cell. The fourth step is the comparison of new clusters with other clusters from the process perspective. Here, a process model is discovered for each cluster, and additional characteristics are compared among the clusters. Through the proposed approach, we can obtain clusters of assembly blocks in consideration of the task and work shop, and can analyze the operation characteristics per cluster. Each step is analyzed in greater detail in the following case study report.

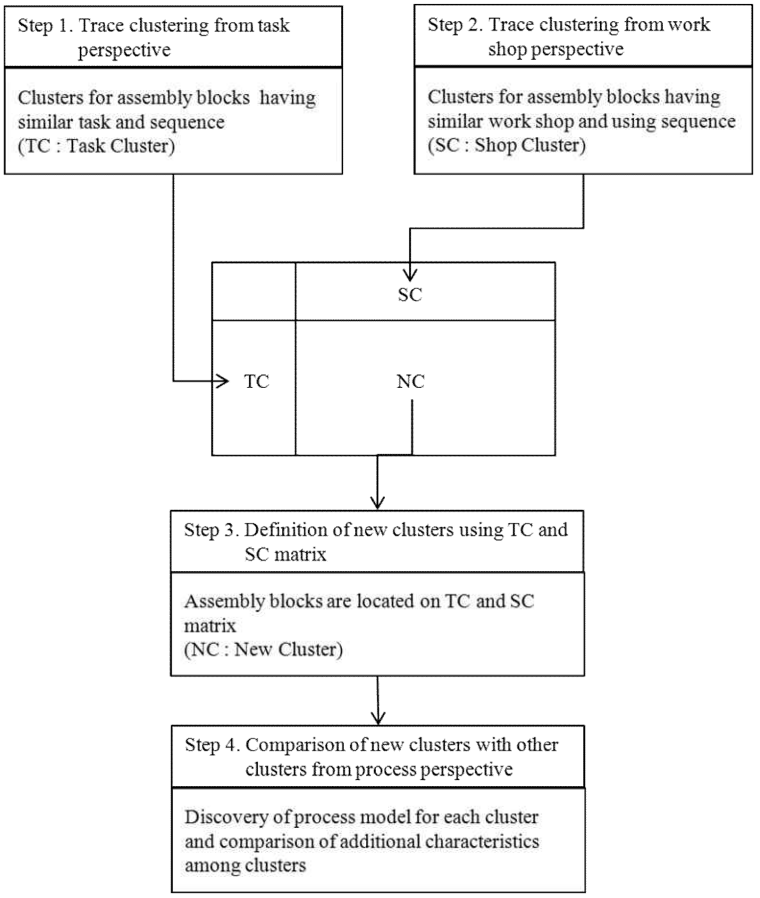



**Fig. 1.** Proposed analysis flow

## 4 Case Study

In this case study, we used two projects' event logs exported from a Block Assembly Monitoring System (BAMS). The data included 135 blocks (i.e., process instances) excepting outsourced blocks. In the data pre-processing step, block data that were not manufactured until final assembly step were removed, and meaningless work stages also were removed at the domain expert's discretion.

### 4.1 Determination of Analysis Data

In the analysis of block assembly operations, three work stages are considered: sub assembly, unit assembly, and grand assembly. Unit assembly is further classified into tree types: small, curved, and large. Thus, five types are considered and expressed in the process model: Sub Assembly (Sub_Assy.), Unit Assembly of Small Type (Unit_Assy.(S)), Unit Assembly of Curved Type (Unit_Assy.(C)), Unit Assembly of

Large Type (Unit_Assy.(L)), and Grand Assembly (Grand_Assy.). However, we also use the work stage codes, because they provide more accurate information. For example, although M1 and M9 both are unit assemblies of the small type, the work level and characteristics are different. Table 1 shows the event log of an assembly block, and Fig. 2 illustrates the assembly structure of the block. In the figure, the five main work stages are indicated by the shaded nodes, and the other work stages are represented uniformly as "Part." It is apparent that many parts are assembled in five main work stages.

In order to perform process mining, attributes have to be mapped into the following process objects among the data fields of an event log: case, activity, time stamp, and originator. The case (also named the process instance) is the ''thing'' that is being handled; for the present case study, "Project" and "Block" were selected as the case. The activity (also named the task, operation, action, or work-item) is an operation performed on the case; in step 1, "Work Stage" and "Work Type" are defined as the activity, and in step 2, "Work Stage" and "Work Shop." For the purposes of our case study, activity was defined differently, in order to obtain clusters from the task and work shop perspectives, respectively. The time stamp, indicating the time of occurrence, is used only at the "Finish Time" for process model analysis. Finally, "Work Organization" is able to be selected as the originator, but it was not considered in this case study because there are so many work organizations. We just inferred the originator through the activity of step 2.

**Table 1.** Event log of assembly block

| Project | Block | Work Stage | Work Type | Work Shop | Finish Time |
|---|---|---|---|---|---|
| 1000 | 101 | C1 | Component | Comp. Shop | 26 May 09:00 |
| 1000 | 101 | SC | Plate | Plate Shop | 30 May 09:00 |
| 1000 | 101 | E1 | Sub Assy. | Assy. Shop 3 | 17 May 17:49 |
| 1000 | 101 | SC | Plate | Plate Shop | 03 June 09:00 |
| 1000 | 101 | P1 | Panel Plate | Plate Shop | 03 June 09:00 |
| 1000 | 101 | M1 | Unit Assy.(S) | Assy. Shop 3 | 13 June 13:00 |
| 1000 | 101 | C1 | Component | Comp. Shop | 08 June 09:00 |
| 1000 | 101 | C5 | Component | Comp. Shop | 15 June 09:00 |
| 1000 | 101 | SC | Plate | Plate Shop | 10 June 09:00 |
| 1000 | 101 | C1 | Component | Comp. Shop | 08 June 09:00 |
| 1000 | 101 | SC | Plate | Plate Shop | 13 June 09:00 |
| 1000 | 101 | P3 | Panel Plate | Plate Shop | 13 June 09:00 |
| 1000 | 101 | P2 | Panel Plate | Plate Shop | 10 June 09:00 |
| 1000 | 101 | H2 | Unit Assy.(L) | Assy. Shop 3 | 21 June 06:42 |
| 1000 | 101 | SC | Plate | Plate Shop | 16 June 09:00 |
| 1000 | 101 | M9 | Unit Assy.(S) | Assy. Shop 3 | 21 June 13:00 |
| 1000 | 101 | S6 | Sub Assy. | Assy. Shop 3 | 26 June 06:47 |
| 1000 | 101 | C5 | Component | Comp. Shop | 25 June 09:00 |
| 1000 | 101 | C1 | Component | Comp. Shop | 20 June 09:00 |
| 1000 | 101 | G9 | Grand Assy. | Assy. Shop 3 | 01 July 02:11 |

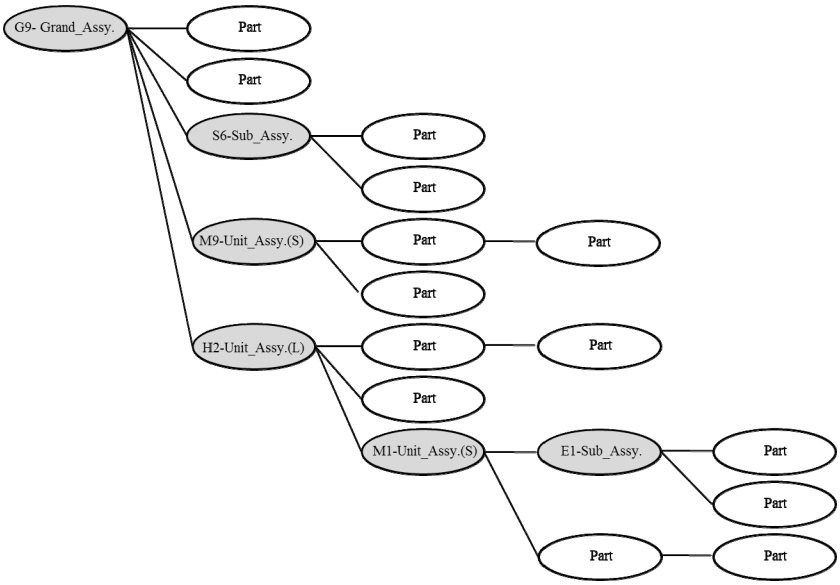



**Fig. 2.** Structure of assembly block

### 4.2 Trace Clustering from Two Perspectives (Step 1 and Step 2)

In step 1 and step 2, trace clustering is performed to cluster assembly blocks having similar operation characteristics. Every clustering algorithm attempts to group sets of similar points, whereas for trace clustering specifically, the points to be clustered are log traces. Traces are characterized by profiles, where a profile is a set of related items that describe the trace from a specific perspective [5].

In this case study, we used activity, activity pattern, and transition profiles. The activity profile defines one item per type of activity (i.e., event name) found in the log. Measuring an activity item is performed simply by counting all of a trace's events that have that activity's name. The activity pattern profile defines items as activity patterns, and calculates the ratio of each activity pattern existing in a trace. The transition profile defines items as directly following the relations of the trace. For any combination of two activity names <*A, B*>, this profile contains an item measuring how often an event with name *A* is directly followed by another event name *B*. This profile is useful for comparing the behaviors of traces.

We applied agglomerative hierarchical clustering with the three profile types. This algorithm gradually generates clusters by merging nearest traces, which is to say, by merging smaller clusters into large ones. The result is illustrated as a dendrogram showing the cluster hierarchy. Our case study used *Euclidean distance* as the distance measure, and *complete linkage* for the clustering distance.

Fig. 3 shows the output of agglomerative hierarchical clustering in step 1. We obtained the seven clusters of task perspectives by setting the cut point to 0.75. In step 2, with the cut point of 0.70, we obtained eight clusters. Overall, for the 135 assembly blocks, we obtained each cluster from the task perspective and work shop perspective.

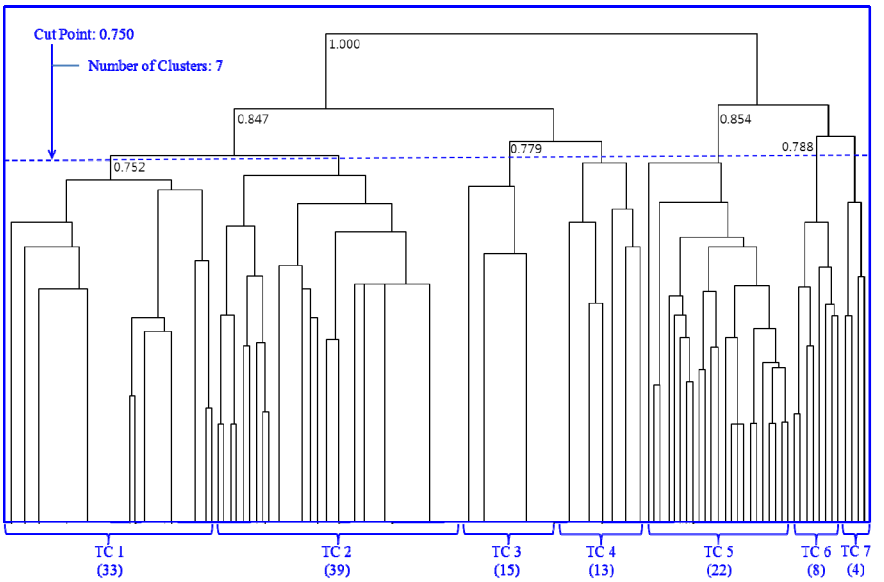



**Fig. 3.** Output of agglomerative hierarchical clustering (clustering from task perspective)

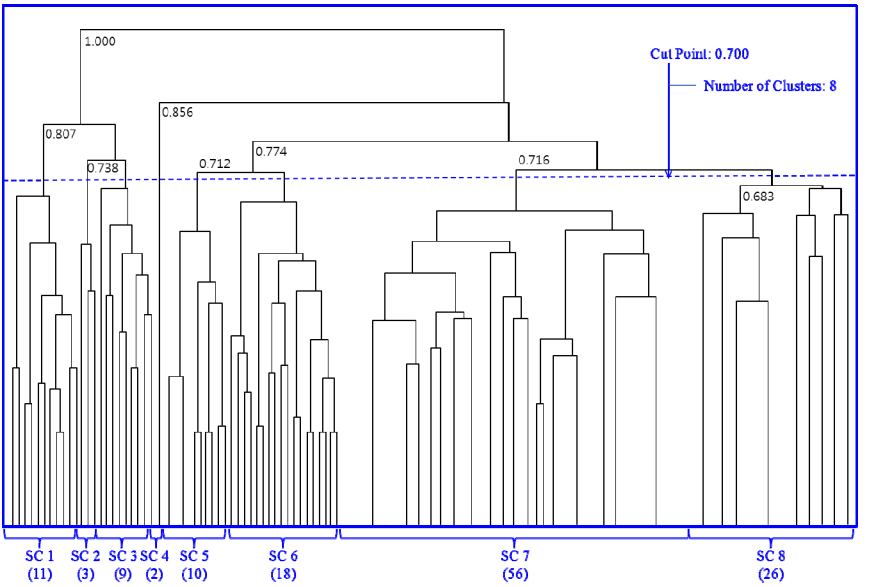



**Fig. 4.** Output of agglomerative hierarchical clustering (clustering from shop perspective)

Each cluster was confirmed using a process model such as sequence diagram or heuristic mining. The characteristics of the clusters are summarized in Table 2 and Table 3, respectively according to perspective. Table 2 displays the output of step 1 (operation characteristics from the task perspective), and Table 3 shows the output of step 2, (operation characteristics from the work shop perspective with respect to block assembly operations). Consequently, each block of the analysis dataset has "TC" and "SC" cluster names through step 1 and step 2. This is used in step 3 to define the new clusters.

**Table 2.** Characteristics of clusters from task perspective

| Cluster | Number of Instances | Characteristics from Task Perspective |
|---|---|---|
| TC 1 | 33 | - Unit Assy.(L) → Grand Assy. |
| TC 2 | 39 | - Specific Sub Assy. → Unit Assy.(L) → Grand Assy. |
| TC 3 | 15 | - Sub Assy. → Grand Assy. |
| TC 4 | 13 | - Unit Assy.(S) → Grand Assy.<br>- Before Unit Assy.(S) is assembled, Sub Assy. is assembled for some cases |
| TC 5 | 23 | - Unit Assy.(C) → Grand Assy.<br>- Before Unit Assy.(C) is assembled, Sub Assy. is assembled for some cases |
| TC 6 | 8 | - Unit Assy.(S) → Unit Assy.(L) → Unit Assy.(C) → Grand Assy. |
| TC 7 | 4 | - Specific Sub Assy. → Specific Unit Assy.(S) → Unit Assy.(C) → Grand Assy. |

**Table 3.** Characteristics of clusters from work shop perspective

| Cluster | Number of Instances | Characteristics from Work Shop Perspective |
|---|---|---|
| SC 1 | 11 | - Most work stages are performed in Shop 4 |
| SC 2 | 3 | - Most work stages are performed in Shop 2 |
| SC 3 | 9 | - Mainly Unit Assy.(C) & Grand Assy. are performed in Shop 2<br>- Previous work stages are performed in other shops for some blocks |

**Table 3.** (*continued*)

| | | |
|---|---|---|
| SC 4 | 2 | - All work stages are performed in Shop 5 |
| SC 5 | 10 | - Most Unit & Grand Assy. are performed in Shop 5<br>- Some blocks are assembled in other shops for Unit Assy. |
| SC 6 | 18 | - Some blocks are finally assembled in Shop 5, after previous work stages are performed in other shops<br>- Some blocks are finally assembled in Shop 3, after previous operations are performed in other shops |
| SC 7 | 56 | - Most work stages are performed in Shop 1 |
| SC 8 | 26 | - In each of Shop 2, Shop 3 and Shop 4, blocks are assembled<br>- Each block has relatively few work stages |

### 4.3 Definition of New Clusters (Step 3)

In step 3, new clusters are defined by the TC/SC matrix. We can assign assembly blocks to the matrix, because the clusters of task perspective and work shop perspective are determined, respectively, in step 1 and step 2. In Table 4's 7 x 8 matrix, assembly blocks are assigned to 19 cells. Correspondingly, 19 new clusters are defined, "NC 1" to "NC 19," ordered sequentially by number of blocks. The most numerous are assembly blocks, having the operation characteristics "TC 2" and "SC 7" simultaneously. That cluster is defined "NC 1". The characteristics are explained in Tables 2 and 3. "NC 1" has the characteristics according to which the operations of unit assembly and grand assembly are performed in assembly shop 1, and the specific sub assembly is assembled before the unit assembly is assembled. This matrix is useful for explaining the relations of task and work shop. When a production plan is formulated for new production not yet manufactured, we can use this matrix to determine the work shop by the characteristics of tasks, or to confirm the tasks assembled by that work shop.

**Table 4.** Definition of new clusters using TC/SC matrix

| Cluster | SC 1 | SC 2 | SC 3 | SC 4 | SC 5 | SC 6 | SC 7 | SC 8 | Sum |
|---|---|---|---|---|---|---|---|---|---|
| TC 1 | | | | 2 (NC 11) | 4 (NC 8) | 2 (NC 12) | 25 (NC 2) | | 33 |
| TC 2 | | | 2 (NC 13) | | 6 (NC 7) | | 31 (NC 1) | | 39 |
| TC 3 | 1 (NC 18) | | | | | | | 14 (NC 3) | 15 |

**Table 4.** (*continued*)

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| TC 4 | | | | | | 2 (NC 14) | | 12 (NC 4) | 14 |
| TC 5 | 8 (NC 6) | 2 (NC 15) | | | | 12 (NC 5) | | | 22 |
| TC 6 | 2 (NC 16) | | 4 (NC 9) | | | 2 (NC 17) | | | 8 |
| TC 7 | | 1 (NC 19) | 3 (NC 10) | | | | | | 4 |
| Sum | 11 | 3 | 9 | 2 | 10 | 18 | 56 | 26 | 135 |

### 4.4 Comparison of New Clusters with Other Clusters (Step 4)

In step 4, the discovered new clusters are compared with other clusters, and additional analysis is performed per cluster. Table 5 shows the relations of the new clusters (NC) with the plan clusters (PC). The plan clusters are the group of assembly blocks managed by production schedulers. They are used to assign the work shop and production duration when a production plan is formulated. But plan clusters do not cover all blocks, which are defined by the production scheduler case by case. So, we need to determine assembly block clusters based on actual data exported from a monitoring system. As shown in Table 5, the most assembly blocks are involved in "PC 12" from the plan cluster viewpoint. In the discovered new cluster, the blocks of "PC 12" are classified into "NC 1" and "NC 2," and "NC 1" also involves the assembly blocks of "PC 11". Namely, 56 blocks is clustered differently by the plan cluster or the new cluster perspective. So, we performed additional analysis for "NC 1," "NC 2," "PC 11," and "PC 12" using the process model.

Fig. 5 and Fig. 6 show the process model after heuristic mining for new clusters "NC 1" and "NC 2," respectively. Fig. 7 and Fig. 8 are the process models of clusters "PC 11" and "PC 12." We can find the preferred process model and cluster with the same analysis data. For example, with respect to the "NC 1," "NC 2," "PC 11" and "PC 12" clusters, the new cluster (NC) is classified to 23 blocks and 25 blocks, whereas the plan cluster (PC) is classified to 8 blocks and 48 blocks.

As shown in Fig. 5, the process model for "NC 1" is better than "PC 11" and "PC 12" from the viewpoint of describing task characteristics that specific Sub Assembly operation is needed before Grand Assembly operation. Also, the process model of "PC 12" is more complicated than that of "NC 1" or "NC 2;" thus it is easier to understand the process model of a new cluster.

**Table 5.** Comparison of new clusters (NC) with scheduler-defined plan clusters (PC)

| Cluster | PC 1 | PC 2 | PC 3 | PC 4 | PC 5 | PC 6 | PC 7 | PC 8 | PC 9 | PC 10 | PC 11 | PC 12 | Sum |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NC 1 | | | | | | | | | | | 8 | 23 | 31 |
| NC 2 | | | | | | | | | | | | 25 | 25 |
| NC 3 | | | | 12 | | | 2 | | | | | | 14 |
| NC 4 | | | | 4 | | 2 | | | | 6 | | | 12 |
| NC 5 | | | 2 | | 2 | | | 4 | 4 | | | | 12 |
| NC 6 | | | | | | | | | 2 | 6 | | | 8 |
| NC 7 | | | | | | | | 6 | | | | | 6 |
| NC 8 | | | | | | 4 | | | | | | | 4 |
| NC 9 | | 4 | | | | | | | | | | | 4 |
| NC 10 | | 3 | | | | | | | | | | | 3 |
| NC 11 | | | | | | | | 2 | | | | | 2 |
| NC 12 | | | | | | | | | 2 | | | | 2 |
| NC 13 | 2 | | | | | | | | | | | | 2 |
| NC 14 | | | 2 | | | | | | | | | | 2 |
| NC 15 | 2 | | | | | | | | | | | | 2 |
| NC 16 | | | | | | | | | | 2 | | | 2 |
| NC 17 | | | | | 2 | | | | | | | | 2 |
| NC 18 | | | | | | | | | | 1 | | | 1 |
| NC 19 | | 1 | | | | | | | | | | | 1 |
| Sum | 4 | 8 | 4 | 16 | 4 | 6 | 2 | 12 | 8 | 15 | 8 | 48 | 135 |

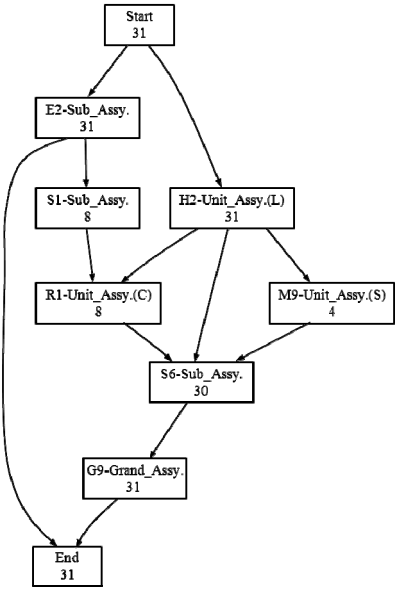



**Fig. 5.** Process model of cluster NC 1

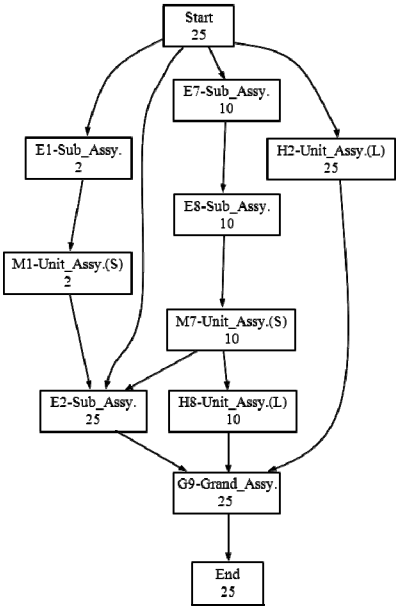



**Fig. 6.** Process model of cluster NC 2

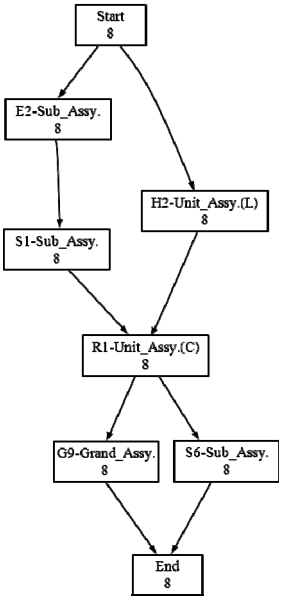



**Fig. 7.** Process model of cluster PC 11

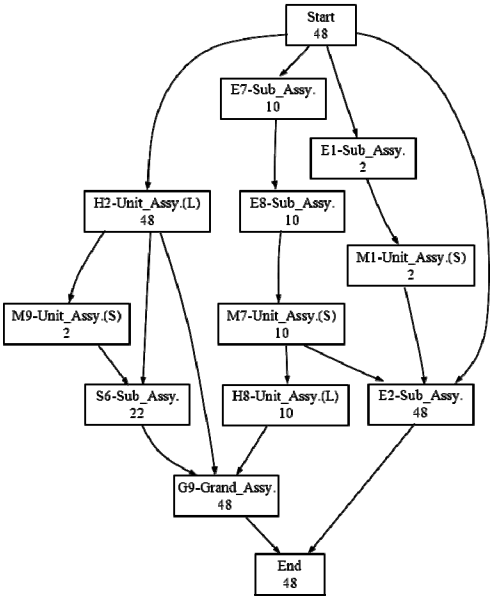



**Fig. 8.** Process model of cluster PC 12

In this step, we also calculate the fitness of the process model in order to confirm the conformance. Table 6 shows the new cluster (NC) and plan cluster (PC) fitness values. That is calculated after the output of heuristic mining transfers to Petri-net.

The fitness of a case with trace $\sigma$ on network $N$ is calculated as following equation (1).

$$fitness(\sigma, N) = \frac{1}{2}\left(1 - \frac{m}{c}\right) + \frac{1}{2}\left(1 - \frac{r}{p}\right) \quad (1)$$

(*c: consumed tokens, m: missing tokens, p: produced tokens, r: remaining tokens*)

The NC fitness is higher than the PC fitness in Table 6. So, we can say that the NC process model explains the assembly operation characteristics well. So we can replace old cluster with new. That also has the better characteristics in the relation between task and work shop.

**Table 6.** Fitness values of new clusters (NC) and plan clusters (PC)

| New Cluster | No. of Instances | Fitness | Plan Cluster | No. of Instances | Fitness |
|---|---|---|---|---|---|
| NC1 | 31 | 0.975 | PC11 | 8 | 0.977 |
| NC2 | 25 | 0.946 | PC12 | 48 | 0.939 |
| | 56 (Sum) | 0.961(Avg.) | | 56 (Sum) | 0.958(Avg.) |

## 5 Conclusions

This paper proposes a new, clustering and operation analysis method for assembly blocks based on process mining techniques suitable for the shipbuilding industry. The new approach consists of four steps: 1) trace clustering from the task perspective, 2) trace clustering from the work shop perspective, 3) definition of new clusters considering task and work shop simultaneously, and 4) comparison of new clusters with other clusters from the process perspective. Clustering and operation analysis output can be used for production planning purposes such as resource allocation and operation scheduling. The effectiveness of the proposed method was verified in a case study using real event logs generated from the Block Assembly Monitoring System (BAMS), an information system. A high priority for forthcoming research will be a detailed comparison of actual and planned processes.

**Acknowledgments.** This study was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MEST) (No. 2012R1A1A2008335).

## References

1. van der Aalst, W.M.P.: Business alignment: using process mining as a tool for Delta analysis and conformance testing. Requirements Engineering 10, 198–211 (2005)
2. Goedertier, S., de Weerdt, J., Martens, D., Vanthienen, J., Baesens, B.: Process discovery in event logs: An application in the telecom industry. Applied Soft Computing 11, 1697–1710 (2011)
3. Lee, S., Kim, B., Huh, M., Cho, S., Park, S., Lee, D.: Mining transportation logs for understanding the after-assembly block manufacturing process in the shipbuilding industry. Expert Systems with Applications 40(1), 83–95 (2013)
4. Lee, D., Bae, H.: Analysis framework using process mining for block movement process in shipyards. ICIC Express Letters 7(6), 1913–1917 (2013)
5. Song, M., Günther, C.W., van der Aalst, W.M.P.: Trace clustering in process mining. In: Ardagna, D., Mecella, M., Yang, J. (eds.) Business Process Management Workshops. LNBIP, vol. 17, pp. 109–120. Springer, Heidelberg (2009)
6. de Weerdt, J., Schupp, A., Vanderloock, A., Baesens, B.: Process Mining for the multi-faceted analysis of business processes - A case study in a financial services organization. Computer in Industry 64, 57–67 (2013)

# DTMiner: A Tool for Decision Making Based on Historical Process Data

Josue Obregon, Aekyung Kim, and Jae-Yoon Jung

Dept. of Industrial and Management Systems Engineering, Kyung Hee University,
1 Seochen-dong, Giheung-gu, Yongin, Gyeonggi, Republic of Korea
{jobregon,akim1007,jyjung}@khu.ac.kr

**Abstract.** Process mining is a discipline that uses techniques to extract knowledge from event logs recorded by information systems in most companies these days. Among main perspectives of process mining, organizational and time perspectives focus on information about resources stored on the event logs and timing and frequency of the events, respectively. In this paper we introduce a method that combines organizational and time perspectives of process mining with a decision support tool called decision trees. The method takes the information of historical process data by means of an event log, generates a decision tree, annotates the decision tree with processing times, and recommends the best performer for a given running instance of the process. We finally illustrate the method through several experiments using a developed plug-in for the process mining framework ProM, first using synthetic data and then using a real-life event log.

**Keywords:** process mining tool, decision tree, decision making, recommendation.

## 1 Introduction

Data recorded by information systems are increasing in today's business environment allowing business analysis tools, which use this data to work, gain more and more value every day. One of these tools is process mining. The idea of process mining is to extract knowledge from the so-called event logs and discover, monitor and improve real processes. Process mining has three types of functions: discovery, conformance and enhancement. Discovery techniques take an event log as input and generate a process model as output using a plethora of notations like petri nets, causal networks, heuristic networks, and so on. Conformance techniques take an existing process model and compare with an event log in order to detect, locate, explain and measure deviations between the model and the actual execution of the process. Enhancement techniques extend or improve process models based on the information obtained the event log. Among different perspectives of process mining, in this paper we focus on two of them, the organizational perspective and the time perspective. Organizational perspective deals with the resource attributes of the event log (e.g., performers of activities), while time perspective considers timing and frequency of events (e.g. processing time of an activity) [1]. On the other hand, decision trees is a

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-319-02922-1_10

M. Song, M.T. Wynn, and J. Liu (Eds.): AP-BPM 2013, LNBIP 159, pp. 81–91, 2013.

decision-making tool that helps to clarify for management the choices, risks, objectives, monetary gains and information needs involved in an investment problem [2].

In this paper we take the method developed in [3] and verify its applicability by means of experiments using two kinds of data. The first experiment is conducted with synthetic data related with a repair process used in [4]. The second experiment is conducted with real-life data. Each experiment is accompanied with performance measures in order to evaluate its accuracy. The experiments are conducted using a developed plugin for the process mining framework ProM called *DTMiner*.

The remainder of this paper is organized as follows. Related work is discussed in Section 2. Section 3 introduces the technique for constructing decision trees based on historical process data. Section 4 presents the implemented plug-in *DTMiner*. Section 5 presents the conducted experiments. Section 6 shows the results of the experiments and Section 7 discusses limitations, recommendations and conclusions of the paper.

## 2 Related Work

Process mining has proved its applicability in real life cases. In [5] a case study illustrating the practical application of process mining is presented. The authors pointed out that the case study showed that it is worthwhile to combine different mining perspectives to reach a richer understanding of the process. The method used in this paper also combines two perspectives of process mining, organizational and time perspectives.

Furthermore, in [6] a semi-automatic approach intended to reduce the number of manual staff assignment is described. Their approach applies a supervised machine learning algorithm to the process event log in order to learn the activities that each performer undertakes. Experiments on three enterprises' datasets were conducted and good overall prediction accuracy was achieved, reaching over 75%. In the technique used in this paper [3], process mining is not combined with machine learning algorithms, instead of that a decision support tool called decision trees is utilized and a simple algorithm for constructing the decision tree is used.

Another works [7, 8] also use machine learning approaches combined with process mining to achieve their results related with staff assignment and decision mining, respectively. In [7] they showed that the problem of deriving staff assignment rules using information from historical process data and organizational information can be interpreted as an inductive learning problem, therefore they used decision tree learning to derive meaningful staff assignment rules. In [8], a plug-in called *Decision Miner* that analyzes the choice constructs of a petri net process model in the context of the ProM framework was presented. Their approach converts every decision point within the process model into a classification problem, and then they solved that problem using decision tree learning.

It is important to remark that decision tree learning in the machine learning area is different from decision trees as a decision support tool. In [9] it is defined that decision tree learning (i.e., machine learning perspective) provides a powerful formalism for representing comprehensible and accurate classifiers, whereas in [2] it is stated that decision tree (i.e., decision analysis perspective) is a decision-making

tool that helps to clarify for management the choices, risks, objectives, monetary gains and information needs involved in an investment problem.

## 3 Performer Recommendation Using Process Mining

In this section we describe briefly the overall procedure used in [3]. The procedure of the proposed method is represented as shown in Fig. 1. In the first stage, a process model is discovered by process mining tools such as ProM, and a running case can also be observed. Then a decision tree is constructed based on the discovered process model and the event log. From the historical data, a key performance indicator (KPI) can be predicted, and information of performance prediction is projected onto the constructed decision trees. For example, the predicted completion time and cost of each pending tasks are annotated for performers.

In the second stage, a running case is matched with the constructed decision tree. To do that, the decision tree is simplified through filtering to reflect characteristics of process, and an observed running case is then matched to the decision tree. Finally, several subtrees can be extracted and merged by matching.

In the last stage, we finally recommend proper performers of each task. Performers are evaluated in terms of time and cost. We can recommend the best performers of scheduled tasks to improve a target KPI.

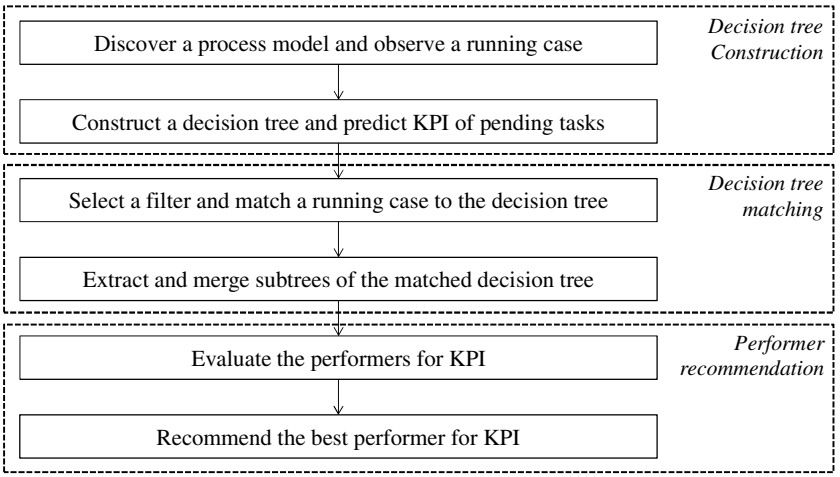



**Fig. 1.** Overview of performance recommendation based on historical data

## 4 DTMiner Plug-in

The technique presented in [3] was implemented as a plug-in for the ProM Framework. The ProM framework integrates the functionality of several existing process mining tools and provides additional process mining plug-ins [10, 11].

Furthermore ProM version 6 offers a new redesigned standard development environment, an enhanced architecture and the user interface that supports new developments on the process mining research area in a relatively easy way. ProM has five kinds of plug-ins, which implement different process mining related functions: mining, export, import, analysis and conversion. We center our attention on mining and analysis plug-ins. Mining plug-ins implement some mining algorithm, e.g., α-miner that constructs a Petri net based on some event log whereas analysis plug-ins implement some property analysis on some mining result[10].

The plug-in called *DTMiner* can be considered as a combination between mining and analysis plug-ins. The *DTMiner* plug-in constructs a decision tree based on an historical process data. In Fig. 2, a generated decision tree is depicted on the main screen of the plug-in interface. Decision nodes are colored with blue and have square shape meanwhile chance nodes are colored with green and have ellipse shape. Decision nodes represent tasks and chance nodes represent performers extracted from the event log. Node information is displayed when the mouse pointer is over the node and it varies depending on the type of it. If it is a decision node, remaining time and route are displayed and if it is a chance node, average time and frequency are displayed. The edge connection between chance nodes and decision nodes displays the average task time taken by the performer to finish the previous task. The underlying decision tree model used for the construction of the decision model stores all the information obtained from the event log. Each chance node is annotated with start and finish task times for each case of the performer that it represents as well as the case frequencies.

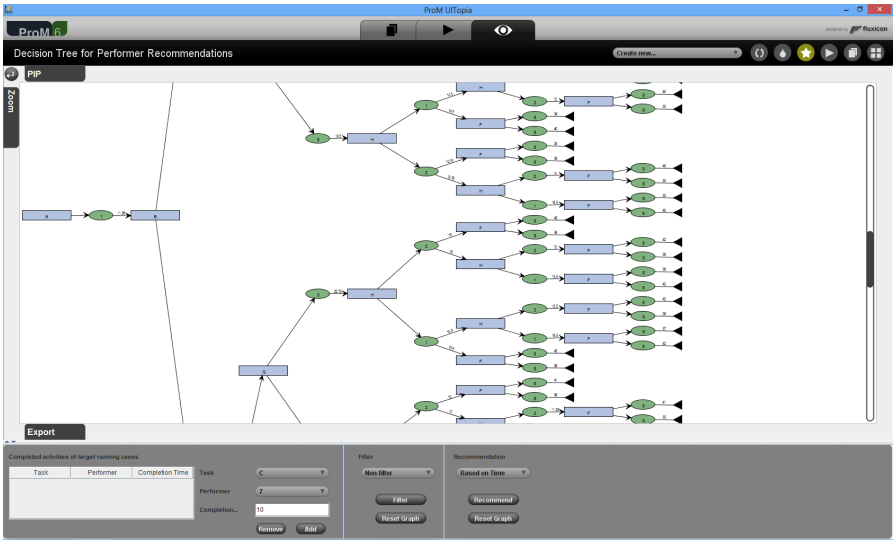



**Fig. 2.** Screenshot of *DTMiner* plug-in showing some results of test data

After loading the event log and generating the decision tree, one can analyze the resultant graph using the analysis section of the plug-in. The analysis section can be visualized at the bottom of the Fig. 3. It has three sections. In the first section, completed activities of target running cases can be added or deleted. In the second section, a filter to match the tree with the target running case can be selected. Finally on the third option a recommendation is given depending the parameter selected.

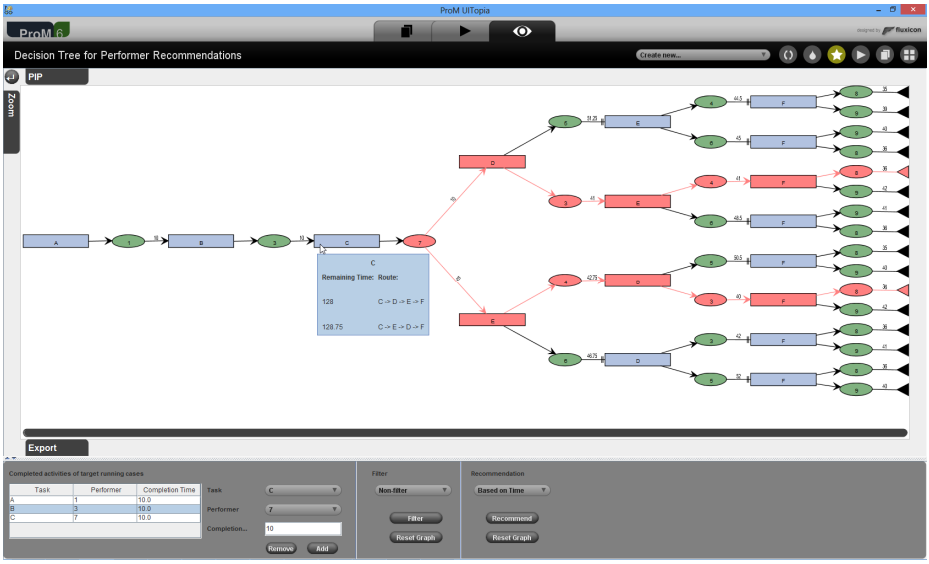



**Fig. 3.** Screenshot of *DTMiner* plug-in showing possible routes from the last task of the running instance and the recommended performers per route

Fig. 3 shows an example already filtered and analyzed. Using the filter non-filter the initial decision tree was pruned. After this, a recommendation based on remaining time is given. When the mouse pointer is over a task node, the possible routes from that point until the end are displayed. The remaining time for each route is also displayed beside the corresponding route. Recommended routes (i.e., nodes and arcs) are colored with red color whereas the arcs of the routes that are not recommended have two perpendicular lines indicating that are blocked.

In the next sections, the *DTMiner* plug-in is used as a proof-of-concept implementation over several event logs.

## 5 Experiments

In this section, we demonstrate the applicability of our approach using one synthetic event log obtained via ProM and a real-life log used in a case study. For the case study we analyzed a process in Dutch Financial Institute.

### 5.1 Synthetic Example

For the first experiment we use an event log about a process of repairing telephones in a company. In Fig. 4, we can see that the process starts by registering a telephone device sent by a customer. After registration, the telephone is analyzed and its defect is categorized. Once the problem is identified, the telephone is sent to the Repair department. The Repair department can fix simple defects and complex defects. Once a repair employee finishes working on a phone, this device is sent to the Quality Assurance department. Then the phone is analyzed by an employee to check if the defect was indeed fixed or not. If the defect is not repaired, the telephone is again sent to the Repair department. If the telephone is repaired correctly, the case is archived and the telephone is delivered to the customer.

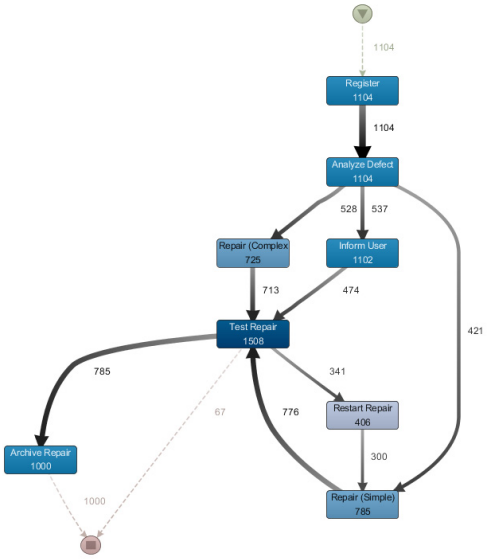



**Fig. 4.** Telephone repair process discovered by the improved fuzzy miner in Disco

In Fig. 4, it can be noted that 1,104 cases exist in the event log and begin with *Register* activity. Among those cases, just 1,000 cases have finished. We used cross-validation in our experiment. Cross-validation is the statistical practice of partitioning a sample data set into two subsets, training set and test set. Training set is used to analyze the data while testing set is used for validation. Because of the nature of the plug-in, in which every case should be tested by hand, our test set had a size of 10 cases and was selected randomly.

The experiment was conducted as follows. Take the real case from the test data, record the actual completion time and the performer. Use the plug-in and enter the first two activities as a running case and get the recommendation. After this, record the new recommended time and check if performers are different from the performers that actually executed the task on the test case. Repeat this for every case in the test data.

The results are summarized in Table 1. It is clear that the method works and always recommends the performer who has registered the shortest average time on the training data. This has a limitation that will be discussed later, about the fact that the recommended performer might be busy at the time when the running case is being executed.

**Table 1.** Summary of experiment results for synthetic data

| Case | Total remaining time (min) | Recommended time (min) | Difference (min) | Number of performers changed |
|---|---|---|---|---|
| 1 | 47 | 8.5 | 38.5 | 0 |
| 2 | 51 | 11.31 | 39.69 | 1 |
| 3 | 28 | 11.31 | 16.69 | 2 |
| 4 | 58 | 11.31 | 46.69 | 1 |
| 5 | 55 | 11.31 | 43.69 | 1 |
| 6 | 21 | 11.31 | 9.69 | 2 |
| 7 | 23 | 12.84 | 10.16 | 2 |
| 8 | 27 | 14.31 | 12.69 | 2 |
| 9 | 23 | 12.26 | 10.74 | 1 |
| 10 | 19 | 6.75 | 12.25 | 2 |

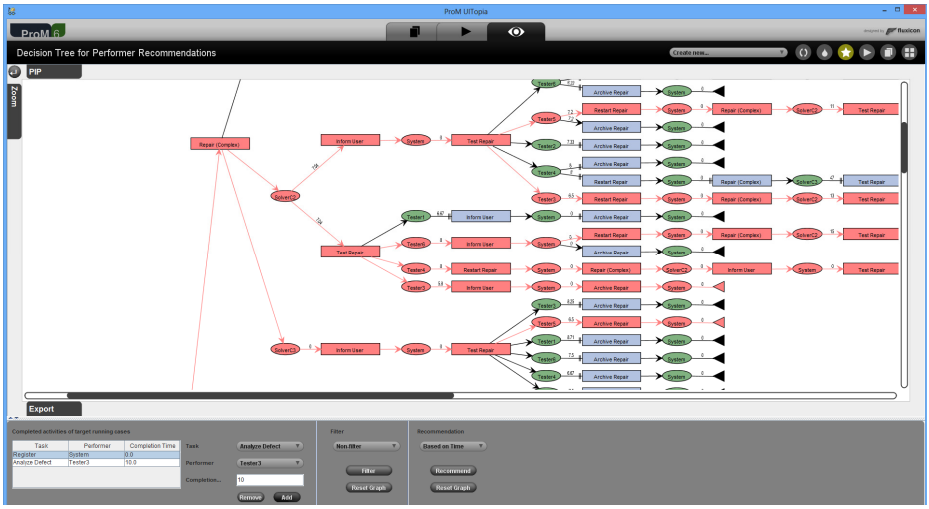



**Fig. 5.** Screenshot of *DTMiner* plug-in showing possible routes from the last task of a running instance and the recommended performers per route

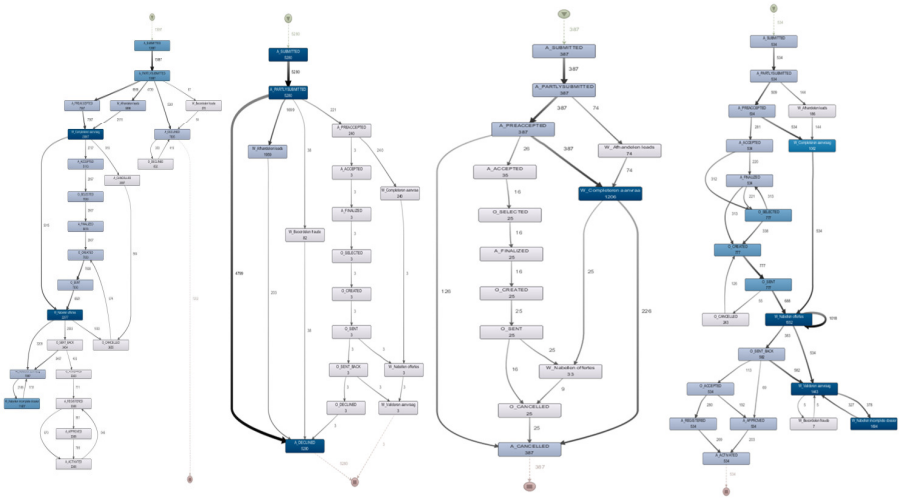
## 5.2 Case Study

We also evaluated the proposed approach using an event log from the Dutch Financial Institute. This log contains 13,087 cases and 262,200 events over a six month period from October 2011 to March 2012. The process represented in the event log is an

application process for a personal loan or overdraft within a global financing organization.

An incomplete case means unexpected case appearing because of extracting data from a particular period of time. Since information systems record events continuously, the log contains some cases which have not finished yet. To get rid of incomplete cases and provide some insight into the structure of the process, we used Disco which draws process models using the improved fuzzy algorithm. It also shows meaningful information such as variants, frequency, and duration and provides powerful filtering features. We found that the whole process can be split into three sub-processes by end events (i.e. *A_DECLINED, A_CANCELLED, A_ACTIVATED*). In the next subsection, we use the three groups split from whole cases, which contain 7,635, 2,807 and 2,246 cases, respectively.

There are some events in the log where the resource information is missing. For these reasons, before testing our approach the log needs to be preprocessed. We first removed all the cases which have at least one event with NULL resource information because they cannot be used for the performer recommendation method. Second, we used only cases whose sequence of activities is shared by at least 10 cases using variation filtering functionality of Disco. Moreover, we consolidated all the resources performed in automatic activities which have zero duration into a resource called 'Automatic'. Finally, we split the filtered log into three sub-processes. As a result the group that ends with *A_DECLINED* has 5,280 cases. The *A_CANCELLED* group has 1,024 cases and *A_ACTIVATED* group has 534 cases after filtering. Fig. 6 shows the process models of each group discovered by Disco.



(a) Whole Process (b) *A_DECLINED* (c) *A_CANCELLED* (d) *A_ACTIVATED*

**Fig. 6.** The process models discovered from Dutch Financial Institute's log

We present an example scenario with the log of the Dutch Financial Institute to describe how the proposed approach can be applied to performer allocation problems with the *DTMiner* plug-in. Using *DTMiner* with the example scenario, the historical process log of a business process was analyzed to construct the decision tree, which was used to recommend the best performers for an ongoing instance of the process.

The Dutch Financial Institute would want to reduce the lead time of their services to improve the quality of the customer loan service. They would also want to decrease the cost of their processes. For this reason, the purpose of this experiment is to recommend the best performer who allows the remaining time or the total labor cost to reduce for each next task.

As depicted in Fig. 1 the overall procedure of the proposed approach consists of three primary steps. Following these steps, we first constructed decision trees from the log using the plug-in. In this step, we used three sub logs and constructed decision trees separately.

In the second step, we assume that running case $\sigma_1$ = < (*A_SUBMITTED*, Automatic, 0, 0), (*A_PARTLYSUBMITTED*, Automatic, 0, 0), (*W_Afhandelen lead*, 10913, 3.9, 8), (A_PREACCEPTED, Automatic, 0, 0) > has been captured by the information system. Also, we suppose that a manager does not want to filter with previous performers, and he wants to obtain the recommendation of performers who can reduce the remaining time. We then set up the running case and filter options as shown in the bottom of Fig. 7.

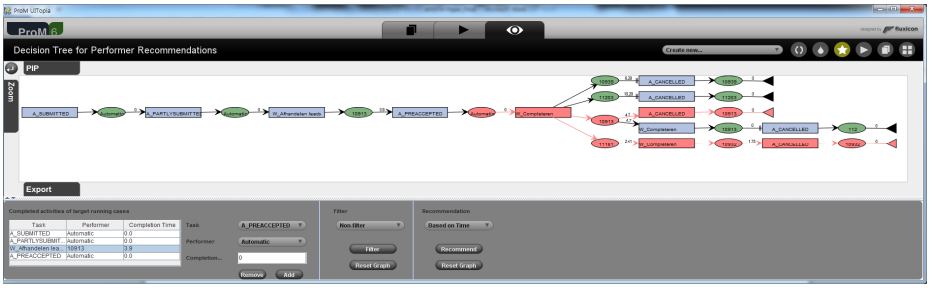



**Fig. 7.** Performer evaluation and recommendation for the sub-process that ends with '*A_CANCELLED*' with a running case $\sigma_1$

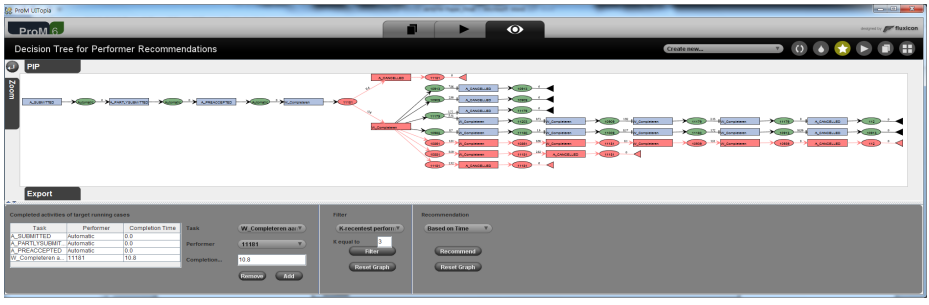



**Fig. 8.** Performer evaluation and recommendation for the sub-process that ends with '*A_CANCELLED*' with a running case $\sigma_2$

In the last step, information about the running case was matched with the decision tree and its subtrees were extracted and merged. Also, the predicted KPIs were updated. Finally, we evaluated performance and recommended the best performer for each next task by reducting inferior performers from leaf nodes. Fig. 7 shows the pruned decision tree and the best performer of each task in sub-process that ends with '*A_CANCELLED* ' for $\sigma_1$. After executing the running case $\sigma_1$, the pruned decision tree showed two possible traces with different excution probabilities as shown in Fig. 7. The first trace $c_1$ =< *A_SUBMITTED*, *A_PARTLYSUBMITTED*, *W_Afhandelen lead*, *A_PREACCEPTED*, *W_Completeren aanvraag*, *A_CANCELLED* > had an execution probability of 40% and the second trace $c_2$ = < *A_SUBMITTED*, *A_PARTLYSUBMITTED*, *W_Afhandelen lead*, *A_PREACCEPTED*, *W_Completeren aanvraag*, *W_Completeren aanvraag*, *A_CANCELLED* > had an execution probability of 60%. Based on this probabilities, we can recommend the best performer for task '*W_Completeren aanvraag*' is '10913' in $c_1$ of which the remaining time is 4.7 and is also the minimum remaining time. In the same way, the best performer for task '*W_Completeren aanvraag*' is '11181' and the best performer for task '*A_CANCELLED*' is '10932' in $c_2$. Also, Fig. 8 shows performer evaluation and recommendation for the sub-process that ends with '*A_CANCLLED*' when a running case $\sigma_2$ = < (*A_SUBMITTED*, Automatic, 0, 0), (*A_PARTLYSUBMITTED*, Automatic, 0, 0), (*A_PREACCEPTED*, Automatic, 0, 0), (*W_Completeren aanvraag*, 11181,10.8, 9) > is given and *3-recent filter* is selected.

## 6 Discussion and Conclusion

In this paper we introduced a tool for decision making based on historical process data. *DTMiner* is a combination between process mining principles and decision trees as a support decision tool. A decision tree is constructed based on an event log, and the decision tree is then annotated with activity processing times that later are used to recommend best performers based on some criteria. Two experiments were conducted with synthetic event log and real-life event log. Through the experiments we illustrated how the method can be applied to recommend good performers.

Some potential limitations still remain in the proposed approach. One limitation comes from the experimentation. Although the performance measures proved that the recommended performer can reduce the final completion time of the process instance, one cannot know if the recommender performer will be available at that moment. In the case is not available, the method should take in consideration waiting time until the performer is not busy anymore, or give an alternative recommended performer.

Another limitation is related with the notion of *completeness* in process mining [1]. One cannot assume to have seen all possibilities in the historical process data used to construct decision trees. If a running case is being evaluated and the sequence of activities was not recorded in the historical data, the method cannot give a recommendation because the branch which refers to the running case does not exist in the constructed decision tree. One way to overcome this limitation could be the use of process models when the decision tree is being constructed, adding possible behavior that actually does not occur but is still possible because of the process model.

**Acknowledgments.** This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (Nos. 2012R1A1B4003505 and 2013R1A2A2A03014718).

## References

1. van der Aalst, W.M.P.: Process mining: Discovery, conformance and enhancement of business processes. Springer, Heidelberg (2011)
2. Magee, J.F.: Decision trees for decision making. Harvard Bus. Rev. 42(4), 126–138 (1964)
3. Kim, A., Jung, J.-Y.: A process mining technique for performer recommendation using decision tree. In: Korean Institute of Industrial Engineers Conference (2012)
4. De Medeiros, A.K.A., Weijters, A.J.M.M.: ProM Framework Tutorial. TechnischeUniversiteit Eindhoven, The Netherlands (2009)
5. van der Aalst, W.M.P., Reijers, H.A., Weijters, A.J.M.M., van Dongen, B.F., Alves de Medeiros, A.K., Song, M., Verbeek, H.M.W.: Business process mining: an industrial application. Inform. Syst. 32(5), 713–732 (2007)
6. Liu, Y., Wang, J., Yang, Y., Sun, J.: A semi-automatic approach for workflow staff assignment. Comput. Ind. 59(5), 463–476 (2008)
7. Ly, L.T., Rinderle, S., Dadam, P., Reichert, M.: Mining staff assignment rules from event-based data. In: Bussler, C.J., Haller, A. (eds.) BPM 2005. LNCS, vol. 3812, pp. 177–190. Springer, Heidelberg (2006)
8. Rozinat, A., van der Aalst, W.M.P.: Decision mining in ProM. In: Dustdar, S., Fiadeiro, J.L., Sheth, A.P. (eds.) BPM 2006. LNCS, vol. 4102, pp. 420–425. Springer, Heidelberg (2006)
9. Quinlan, J.R.: Decision trees and decision-making. IEEE T. Syst. Man Cyb. 20(2), 339–346 (1990)
10. van Dongen, B.F., de Medeiros, A.K.A., Verbeek, H.M.W(E.), Weijters, A.J.M.M.T., van der Aalst, W.M.P.: The ProM framework: Anew era in process mining tool support. In: Ciardo, G., Darondeau, P. (eds.) ICATPN 2005. LNCS, vol. 3536, pp. 444–454. Springer, Heidelberg (2005)
11. Verbeek, H.M.W., Buijs, J.C.A.M., van Dongen, B.F., van der Aalst, W.M.P.: ProM 6: The process mining toolkit. In: BPM Demonstration Track, vol. 615, pp. 34–39 (2010)

# Process Discovery by Synthesizing Activity Proximity and User's Domain Knowledge

Bernardo Nugroho Yahya[1,*], Hyerim Bae[2], Sung-ook Sul[3], and Jei-Zheng Wu[4]

[1] Ulsan National Institute of Science and Technology,
UNIST-gil 50, Eonyang-eup, Ulju-gun,
689-798 Ulsan, South Korea
[2] Industrial Engineering Department, Pusan National University,
Busandaehak-ro 63 beongil, Geumjong-gu,
609-735 Busan, South Korea
[3] Total Soft Bank, Ltd.
Hanjin Shipping Building, 79-9, Jungang-dong 4-Ga, Jung-gu,
600-014 Busan, South Korea
[4] Soochow University
56 Kueiyang Street, Section 1,
Taipei 100, Taiwan, R.O.C.
bernardo@unist.ac.kr, hrbae@pusan.ac.kr,
sosul@tsb.co.kr, jzwu@scu.edu.tw

**Abstract.** Process mining techniques assist users to automatically infer process models from event logs. However, the result of process model driven by traditional process mining technique may conflict with the knowledge of users due to some real conditions, i.e. alternative activity is selected due to equipment breakdown. First, the use of heuristics may detect inconsistencies caused by bad guess. Second, extraction of all possible ordering of events reflects historical observation that sometimes hinders users to obtain an ideal process model since the activity has some event types. Yet, the current process mining approach is not totally compatible with some aspects such as extra logs behavior and soundness of process model when the process model changes according to user requirements. This paper presents a method for synthesizing activity proximity from event logs in the area of process mining. The method derives a bounded graph that covers the extra-behavior of an event log according to user's domain knowledge. Another important property is that it produces a graph with considering the proximity among activities that still contains the original behavior of the event log based on event types. The methods described in this paper have been implemented in ProM framework and tested on a set of real process examples.

**Keywords:** process mining, business process, proximity score, user's knowledge, integer linear programming.

* Corresponding author.

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-319-02922-1_10

M. Song, M.T. Wynn, and J. Liu (Eds.): AP-BPM 2013, LNBIP 159, pp. 92–105, 2013.

# 1 Introduction

In many business applications, the primary purpose of discovering a process model is to act as a means of communication such that a process model facilitates the understanding of complex business processes from stakeholders [1]. The stakeholders are likely to employ traditional design approaches, where models are eventually built by refining and formalizing a number of specifications based on their prior knowledge. Although the stakesholders have their understanding about process model, the sequence of tasks performed along several enactments of a transactional system may be different from original process model. As a consequences, by applying process mining as a tool used to discover a process model from event logs, the discovered process model may not reflect the prior knowledge they possess about the process execution since current process discovery techniques can automatically derive a process model based on the statistics hidden in the event logs. As a result, mined model may violate conceptual specifications and domain-constraints. Hence it is required to further pre- or postprocessing for obtaining a better quality of mined model.

Process mining is a tool used to discover a process model from event logs. The existing techniques in process mining can discover process model based on the tasks performed in an enterprise system. However, the mined model, as a result of proces discovery, may show different outcome with user's prior knowledge based on the technique that analyst used. Therefore, it is important to involve expert's knowledge to discover the mined model. Although the traditional mining techniques allows user's intervention to the result of mined models [4][9], their modification technique toward the user requirement is not compatible and somehow unsound. For example, modification of events that occur in sequence to be a parallel behavior can give result to some activities that may not be connected, events that occur in sequence may not represent as adjacent in the process model and it mislead to some activities that may not reach finish properly. Other work, for example, artificial generated negative events (AGNEs) [12], has detailed description of generating artificial negative events rather generating constraint. The generated negated events, which also include log's noise, may possibly lead to false conclusions. The work on process discovery via precedence constraints seems more similar to this proposed approach [11]. However, there is no discussion on soundness property, which is important in the domain of process discovery i.e. expert give the desired constraints and lead to discover unsound process. Moreover, some other issues such as additional constraints (i.e designated start, designated end), which is very critical issue for process discovery, and interface for the convenience of users are not discussed in detail.

This paper proposes a discovery technique, called as proximity miner, which has a goal to discover a process model based on prior knowledge of experts and the behavior of event logs. However, the strategy and contribution are different in two ways. First, it introduces *activity relations* by synthezising activity proximity that guarantee the inclusion of the observed behavior of the event logs in our integer linear programming (ILP) model. Second, the graph produces a sound model with extra behavior in the resulting graph based on user’s background knowledge. By applying

proximity score measurement on the mining technique, our proposed model tends to extract extra behavior with additional constraints to ensure the soundness of the result of process model although there is a change of log relations. The term of extra behavior in this study considers the log behavior that is unable to be retrieved from event logs. In addition, expert can point out the designated start and end activity when a log contains several abstract processes.

This paper is organized as follows. Section 2 introduces the example of our study. Section 3 explains the formal definition of this study by showing how the integer linear programming (ILP) combined with the concept of proximity can extract extra log behavior. Section 4 shows the main features of our approach, its implementation in ProM framework and experiment result. Related works are discussed in section 5 and section 6 concludes the paper.

## 2 Running Example

The starting point of our work was a case study concerned with the behavior of containers in the domain of port logistics. Users roughly understand the behavior of the container which has several types (reefer container, general container and empty container) [8]. Using process discovery, we tried to answer the questions "How do the containers actually flow?", "Do the containers flow correctly?", i.e. based on the event logs, we automatically constructed process model showing the ordering and frequency of container flows. Since the event logs contain outliers of containers' flow, user wanted to put their knowledge to improve the process model quality. Finally, we used process mining to answer the question "How do we get an ideal process model based on the user's knowledge?"

To briefly explain the port logistics process, we described 3 main activities used in landside transport as shown in the left part of Fig. 1. Example of event logs is shown in the right part of Fig. 1. *Discharging by yard crane* activity (A) and (B) describe the process discharging in the yard after an incoming container from the vessel into the port. If the container type is reefer, the port officer needs to ensure that the container can be powered into certain locations to maintain the temperature. Thus, reefer container should go to the activity *Reefer container Plugging* (C) and (D). If a scheduled truck is coming into the port, then the container is ready for the delivery. The reefer container will be plugged off from the power and picked up from the yard for delivery by truck (*Yard crane task for Gate Out* (E) and (F)). If it is a general type of container, it stays in the yard and waits for *Yard crane task for Gate Out* without managing *Reefer container Plugging* activity. This is a typical example of the landside discharging process.

The common meta model (called as event type) for process mining data usually begins with *schedule*, followed by *start* and ends with *complete* [16]. However, current event data recording technology in port depends on human on which sometimes fail to represent the real time, i.e. operators only record completion of activity without recording the *start* event caused by high workload. Thus, port's expert decides to analyze event logs using *schedule* and *complete* event, which may have difference meta model with the original meta model derived from existing mining

techniques. In this event log, the event type *schedule* means that a user receives an inquiry on that activity and schedule the task on the system. The latter example shows the rationale of our approach.

The log's representation in left part of Fig. 1 shows that when the operation time between event A and B is too long, event C, that is possible to be executed after event A and before event B, can occur in between of event A and B. Thus, the log trace does not represent the process model of the port manager's mind. Fig 2(a), 2(b) and 2(c) are the mined model as a result of process mining techniques based on frequency, semantic behavior and noise reduction using frequency abstraction miner, heuristic miner and fuzzy miner, respectively. Commonly, the mining techniques considered the adjacent relationship of events and it results a bad guess of model which is not properly expected by the experts. Process model shown in Fig. 2 (d) is one that is in a port manager's mind. When the analyst shows the mined model result to the port's expert (Fig 2(a)), they pointed out two problems. First, some of the cases do not represent the expert's mind, i.e. A->C, C->B, E->D, D->F. Second, some of the moves in the model does not occur during execution, i.e. C-D. As well as frequency abstraction miner, both heuristic miner and fuzzy miner (by using default parameters) have not represented expert's mind about process mined model. Thus, the participation of experts on mining a process model from logs is necessary. This study will discuss how the experts' participation improve the mined model.

| Event name (event type) | Index |
|---|---|
| *Discharging by yard crane* (schedule) | A |
| *Discharging by yard crane* (complete) | B |
| *Reefer container Plugging* (schedule) | C |
| *Reefer container Plugging* (complete) | D |
| *Yard crane task for Gate Out* (schedule) | E |
| *Yard crane task for Gate Out* (complete) | F |

| CaseID | Log Trace |
|---|---|
| 1 | ACBDEF |
| 2 | ACBEDF |
| 3 | ACBDEF |
| 4 | ABCDEF |

**Fig. 1.** Event logs example (left) and process model using graph (right)

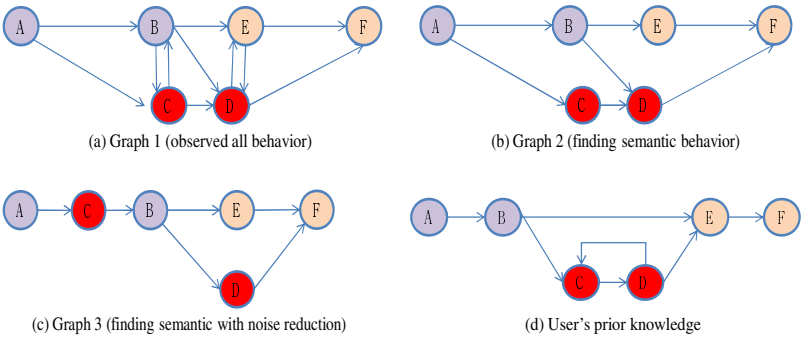



**Fig. 2.** Mined model from event logs with (a) observed all behavior, (b) finding semantic behavior and (c) finding semantic with noise reduction. Figure (d) represents the user's prior knowledge about related process.

## 3 Formal Definition

Commonly, real world application is stored in an event log. An event log, denoted as $L$, consists of a set of process instances or cases, and each case is described by a sequence of events. The sequence of events contains behaviors of activities to indicate the flow of activities from beginning until the end. This section explains the formal definition of example log in section 2.

**Definition 1.** Event Log
An event log is a tuple of $<E, C, S, R, E_S, E_E>$ which is defined as follows:

- **Event.** $E = A \times \{0,1\} \times T$ is a set of events, where $A$ is a set of activities, $T$ is a set of timestamp and 0 means that the underlying event is *schedule* type of activity, and 1 means *complete* type of an activity. To represent activity, type and timestamp of each event, we denote notation as follows: *e.act* refers to the activity name, *e.type* refers to the event type and *e.time* refers to the event timestamp. If *e*=(yardJobLoad, 0, 2012-06-12 01:54:32), then *e.act* = yardJobLoad, *e.type* = SCHEDULE and *e.time* = 2012-06-12 01:54:32. $E(L)$ represents event set in a log $L$.
- **Case.** A case is a set of event for its one instance, i.e. $C = \{c_k \mid k=1,..., K\}$. A case $c_k$ corresponds to the trace $<e^k_1, e^k_2, ..., e^k_n>$ where each $e^k_i$ denotes the $i$-th event in case $k$. $e^k_i \in E$ is an event in a single instance of workflow for $1< i < I^k$ and $I^k$ is the total event in $k$-th case and $\sum_k^K I^k$ is the total events in a log.
- **Direct Event.** Let $L$ be the log of workflow. Direct event set ($S$) is denoted as $s_{ij} \subseteq (e^k_i, e^k_j) \mid e^k_i \succ e^k_j, \forall e^k_i, e^k_j \in E(L)$ where $e^k_j$ is direct post-event of $e^k_i$ (or $e^k_i$ is the pre-event of $e^k_j$) in $k$-th case. The element $e^k_i \succ e^k_j$ represents the fact that $e^k_i$ precedes $e^k_j$ with no any activity $e^k_k \in E(L)$ such that $e^k_i \succ e^k_m$ and $e^k_m \succ e^k_j$. To measure the frequency of direct event set, we denote $F_{ij} = \mid s_{ij} \mid$ where $F_{ij}$ is the number of event $e^k_j$ as the direct post-event of $e^k_i$ for all $k$ in $L$.
- **Traceable Event.** Let $L$ be the log of workflow. Traceable event set ($R$) is denoted as $r_{ij} \subseteq (e^k_i, e^k_j) \mid e^k_i \gg e^k_j, \forall e^k_i, e^k_j \in E(L)$ where $e^k_j$ is reachable from $e^k_i$ (or $e^k_i$ is able to reach $e^k_j$). The element $(e^k_i \gg e^k_j)$ represents the fact that $e^k_i$ precedes $e^k_j$ with certain distances ($d_{ij} \geq 1$). Althought two events has certain distances, we can say that both events has proximity. The concept of proximity will be explained later.
- **Start and End Event.** Let $L$ be the log of workflow.
  1. $E_S(L) \subseteq \{e^k_i \mid s_{ji} = \emptyset, e^k_i \in E(L)\}$ is named as the start event set of the workflow from the log $L$.
  2. $E_E(L) \subseteq \{e^k_i \mid s_{ij} = \emptyset, e^k_i \in E(L)\}$ is named as the end event set of the workflow from the log $L$.

**Definition 2.** Distance
Distance ($d_{ij}{}^k$) is defined as an integer value to represent that two events $e_i^k$ and $e_j^k$ is either direct event (=1) or in between some other events (>1) [19]. Suppose $pa_{ij}$ is denoted as a path from an event $e_i^k$ to another event $e_j^k$ in a case $C$.

• $pa_{ij}^{k}$={$e_i^k$, $e_l^k$, $e_{l+1}^k$, …,$e_{l+m}^k$,…,$e_{l+M}^k$, $e_j^k$}, ($e_i^k$, $e_l^k$)∈$S$, ($e_{l+M}^k$, $e_j^k$) ∈$S$, ($e_{l+m}^k$, $e_{l+m+1}^k$) ∈$S$, ∀$m$

When there is a path $pa_{ij}$ from $e_i$ to $e_j$, the distance between the two events is

$$d_{ij}^{k} = \left| pa_{ij}^{k} \right| - 1 \tag{1}$$

**Definition 3.** Proximity

Proximity is defined as the closeness of two activities. First, we need to calculate the distance between two traceable events. The proximity value, which is denoted by $q_{ij}^k$, is defined as in equation (2). Second, we need to get an average proximity score from all cases. Notation $h_{ij}^k$ equals to 1 if $e_j^k$ is the post-event of $e_i^k$ in $k$-th case, otherwise 0. To gain the average proximity score ($PS_{ij}$) of event $e_i^k$ and $e_j^k$ considering all $K$ cases, we should sum all $q_{ij}^k$ and divide it by $K$. It is measured by the equation (3).

$$q_{ij}^{k} = \frac{h_{ij}^{k}}{d_{ij}^{k}} \tag{2}$$

$$PS_{ij} = \frac{\sum_{k=1}^{K} q_{ij}^{k}}{K} \tag{3}$$

**Definition 4.** Process Model with Graph (Graph-net)

Process model with graph representation (Graph-net) is denoted as tuple ($V$, $ED$) where $V$ is a vertex and $ED$ is an edge to connect $V$ ($ED \subseteq V \times V$ ).

1. $V = A \times \{0,1\}$ represents the event set of activity name and event type. For each event $e_i^k$ with specific event type i.e. schedule, there is a vertex $v_i \in V$.
2. $ED$ represents the edge set. Basically, for each event $e_i^k$ which has direct event $e_j^k$, there is an edge $ed_{ij} = (v_i, v_j) \mid v_i, v_j \in V$, to represent $e_i^k \succ e_j^k$.
3. If there is exist $v_i \in V$ such that $ed_{ji} = \emptyset$, then $v_i$ is a start activity. A set of start activity ($A_S$) is the same with a set of start event ($E_S$)
4. If there is exist $v_i \in V$ such that $ed_{ij} = \emptyset$, then $v_i$ is an end activity. A set of end activity ($A_E$) is the same with a set of end event ($E_E$).

**Definition 5.** Soundness. Soundness is considered as a correctness criterion of a process model [6]. Soundness can be verified using standard Petri-net-based analysis techniques. The following are well-known correctness criterion.

1. safeness: in the use of Petri net, we can say that a place cannot hold multiple tokens at the same time.
2. proper completion: there should be no in-progress activity when the final activity is completed.
3. option to complete: there should always possible to reach final activity.
4. absence of dead tasks: all activities of the model are potentially reachable.

Since we use graph to represent the result of our approach, it is an obvious that *safeness* criterion is not matched. Note that the option to complete implies proper

completion. Thus, we proceed other two criterions as the basis of soundness in our study.

**Definition 6.** Domain Knowledge
There are five definitions of domain knowledge. They are causal dependency, not-related, parallel, designated start and designated end. Domain knowledge are interpreted over graphs as follows.

- **Causal dependency.** Causal dependency is denoted as *CL* where $CL \subseteq \{cl_{ij} = (v_i,v_j) \mid v_i, v_j \in V\}$ is the set of two vertices where $cl_{ij}$ is the existence that two vertices $v_i$ and $v_j$ have causality relationship. ($x_{ij}$=1).
- **Not-related.** Not-related relationship is denoted as *NR* where $NR \subseteq \{nr_{ij} = (v_i,v_j) \mid v_i, v_j \in V\}$ is the set of two vertices where $nr_{ij}$ represent that two vertices $v_i$ and $v_j$ have no relationship. ($x_{ij}$=0).
- **Parallel.** Parallel relationship is denoted as *PL* where $PL \subseteq \{pl_{ij} = (v_i,v_j) \mid v_i, v_j \in V\}$ is the set of two vertices where $pl_{ij}$ is the existence that two vertices $v_i$ and $v_j$ are parallel. The element $(v_i,v_j) \in PL$ represents the fact that $v_i$ parallels with $v_j$. Thus, there is no relation between two vertices $v_i$ and $v_j$ such as $x_{ij}$=0 and $x_{ji}$=0.
- **Designated Start.** Designated start is denoted as $desStart \subseteq \{dst_i = v_i \mid v_i \in V\}$ is the set of vertices that is possible for being a start activity in a process model ($x_{ji}$=0)
- **Designated End.** Designated end is denoted as $desEnd \subseteq \{dse_i = v_i \mid v_i \in V\}$ is the set of vertices that is possible for being an end activity in a process model ($x_{ij}$=0)

Since the concept of proximity includes both traceable and direct events, then it is necessary to measure the benefit and cost of including the relations of two events to find all relevant behaviors. Here, we introduce benefit and penalty as a trade-off score to get the required process model.

**Definition 7.** Benefit and Penalty
In this approach, we introduce *benefit* ($B_{ij}$) and *penalty* ($P_{ij}$) score to represent priority selection when such extra behavior of event *i* and *j* is required. The score between event *i* and *j* for best adjacent selection is necessary for two aspects. First, the measure applies high *benefit* for selecting a direct succession event as adjacent task in process model. Second, it is required not to choose irrelevant behavior based on event logs by giving high value of *penalty*. Since the user can arbitrarily choose the parallel, causality and not-related relationship according to their knowledge, it needs measures to carefully modify current log relations and select the best relations to result a process model. If there exists a link between event *i* and *j* in the event log, then we put value the maximum of $q_{ij}^{k}$ in the benefit and 0 in penalty (since it is our intention to create the process model). On the other hand, we put the value 0 in the *benefit* and the maximum of $PS_{ij}$ in *penalty* to restrict the selection of those event relations which does not exist in the link. The formula to generate the value of *benefit* and *penalty* for all relationships is shown in equation (4) and (5).

$$B_{ij} = \begin{cases} \max_{i,j}(PS_{ij}) & \forall v_i, v_j \wedge F_{ij} > 0 \\ 0 & \forall v_i, v_j \wedge F_{ij} = 0 \end{cases} \tag{4}$$

$$P_{ij} = \begin{cases} \max_{i,j}(PS_{ij}) & \forall v_i, v_j \wedge F_{ij} = 0 \\ 0 & \forall v_i, v_j \wedge F_{ij} > 0 \end{cases} \tag{5}$$

The integer linear programming (ILP) is first proposed to determine a graph model from log. Previous works used *constraint satisfaction problem* as a tool to extract graph model [11]. Others use inductive logic programming as an approach to make negative events [12]. In this study, integer linear programming (ILP) is chosen for a reason to reduce the polynomial time [18], as one problem of complexity in previous approach based on constraint satisfaction problem [11].

$$\max W = \sum (F_{ij} + PS_{ij} + B_{ij} - P_{ij}) * x_{ij} \tag{6}$$

s.t.

$$x_{ij} = 0 \quad \forall (v_i, v_j) \in PL \tag{7}$$

$$x_{ji} = 0 \quad \forall (v_j, v_i) \in PL \tag{8}$$

$$x_{ij} = 0 \quad \forall (v_i, v_j) \in NR \tag{9}$$

$$x_{ij} = 1 \quad \forall (v_i, v_j) \in CL \tag{10}$$

$$\sum_j x_{ij} = 0 \quad \forall v_i \in desEnd \tag{11}$$

$$\sum_j x_{ji} = 0 \quad \forall v_i \in desStart \tag{12}$$

$$\sum_j x_{ij} \geq 1 \quad \forall v_i \notin E_E, \forall v_i \notin desEnd \tag{13}$$

$$\sum_j x_{ji} \geq 1 \quad \forall v_i \notin E_S, \forall v_i \notin desStart \tag{14}$$

$$x_{ij} \in \{0,1\}, \forall (v_i, v_j) \in ED \tag{15}$$

In this study, the objective function attempts to find the maximum value of proximity to mine a graph net from a log (Equation 6). If there is an adjacent relation between event *i* and *j*, then obvious that the value is greater than 0 ($F_{ij} > 0$, $B_{ij} > 0$, $P_{ij}$=0). If there is no adjacent relation between event *i* and *j*, there may be two options.

- First, no adjacent relation but exist traceable relation. the proximity score may have greater value than 0 ($PS_{ij}$ >0, $B_{ij}$ =0, $P_{ij}$ >0) since two activities are reachable in the event logs.
- Second, no adjacent relation and no traceable relation. There is no proximity score ($PS_{ij}$ =0, $B_{ij}$ =0, $P_{ij}$ >0), which is no two traceable events in the logs. Thus, it will return the best adjacent relation according to the given PS value (existing traceable events).

In order to maximize $W$, the value of $P_{ij}$ should be minimized. To discover a process model with minimum penalty ($P_{ij}$=0), a process model should contain only direct event ($F_{ij}$ >0). The value of the penalty will be equal to $\max_k (q_{ij}^k)$ when user's domain knowledge contradict with the existing log behavior, i.e. there is no direct event between two vertices such that $F_{ij}$=0. In the case that $P_{ij}$ is equal to $\max_k (q_{ij}^k)$, then $B_{ij}$ is equal to 0 and it has extra log behavior.

Constraint (7) and (8) show the relations between vertex $v_i$ and $v_j$ should equal to 0 if two events are considered having parallel relation. Constraint (9) presents the relations between vertex $v_i$ and $v_j$ should equal to 0 if two events are considered having not related relation. Constraint (10) describes the relations between vertex $v_i$ and $v_j$ should equal to 1 if two events are considered having causal relation.

Constraint (11) and (12) denote designated start and end activity. Constraint (11) indicates that the relations between vertex $v_i$ and $v_j$ should equal to 0 for all vertices $v_i$ that is a designated end activity ($v_i \in desEnd$). Constraint (12) expresses that the relations between vertex $v_i$ and $v_j$ should equal to 0 for all vertices $v_i$ that is a designated start activity ($e_i \in desStart$)

To obtain a sound process model, it is necessary to build constraints that satisfy the soundness property, as defined in definition 11. For verifying the soundness of process model (option to complete and absence of dead tasks), there should be a constraint to enforce a connection between vertex which neither start nor end event. Constraint (13) shows that the relations between vertex $v_i$ and $v_j$ should be greater or equal to 1 for all vertex $v_i$ that is not an end activity ($v_i \notin A_E$) and a designated end activity ($v_i \notin desEnd$). Meanwhile, constraint (14) defines that the relations between vertex $v_i$ and $v_j$ should be greater or equal to 1 for all vertex $v_i$ that is not a start activity ($v_i \notin A_S$) and a designated start activity ($v_i \notin desStart$). To conform to the requirement of constraints in finding the required objective function, it is necessary to make a decision variable (15). A decision variable, $x_{ij}$, equals to 1 if vertex $v_i$ immediately precedes vertex $v_j$, where $(v_i, v_j) \in ED$; 0, otherwise.

## 4 Implementation

The proximity miner is implemented in ProM process mining framework and demonstrated using port logistics data. Since the workflow model element consists of two types, schedule and complete, each event should be identified by a concatenated key of workflow model element and event type. The mathematical model was implemented using a commercial software LINGO 11.0 and an open source of linear programming tool, LpSolve 5.5.0. It runs on a desktop computer with Core i7 860 2.80 GHz, 8 GB RAM. Fig. 3 shows the main page of proximity miner in the Process Mining Framework, called ProM. Fig. 4 shows the result of proximity miner by using a case study on port logistics process.

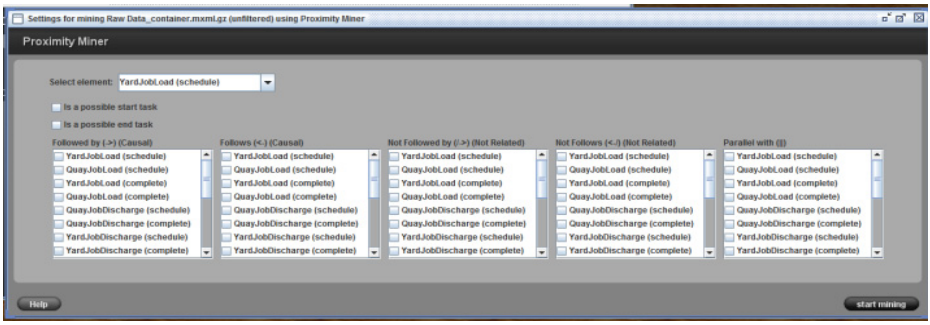



**Fig. 3.** Main page of proximity miner

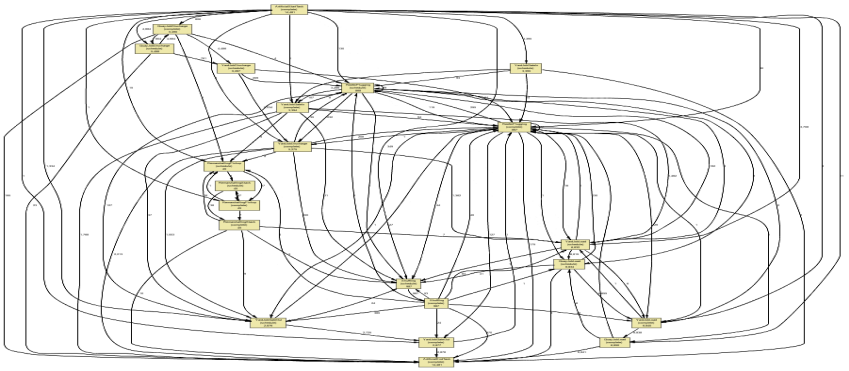


**Fig. 4.** Result of Proximity Miner

Some data experiments (as shown in Table 1) were used to analyze the performance of proximity miner. It shows that proximity miner runs slower when data includes a lot of loop behavior. In the process mining setting, running times for solution approaches do not usually represent a major issue (as algorithms are applied offline, i.e. during the (re)-design phase). Since the main focus of this study is about generating a better quality of the mining result, we put the performance problem as our future work. Container data are used for the experiment with the conformance checking tool. The proposed algorithm is converted into Petri Net and evaluated using existing conformance checker tool. For a comparison with other mining techniques, the experiment is also conducted using container data in alpha miner and heuristic miner (see Table 3).

**Table 1.** Analysis of Port Logistics Process

| Data | Cases | Events | Performance | Note |
|---|---|---|---|---|
| Container | 14,481 | 65,307 | 9 sec 302 ms | - |
| Vessel | 22 | 15,787 | 1 min 50 sec | High frequency of loops |
| Truck | 18 | 21,270 | 1 min 23 sec | High frequency of loops |
| Block | 7624 | 25580 | 2 sec 267 ms | - |
| QuayCrane | 8 | 16211 | 1 min 47 sec | High frequency of loops |
| YardCrane | 24 | 56975 | 8 min 30 sec | High frequency of loops |

In this study, we apply the measures from Rozinat and van der Aalst (2008), fitness and behavioral appropriateness. Fitness (*f*) is a metric that is obtained by trying whether each (grouped) sequence in the event log can be reproduced by the generative model. It is often called as sequence replay. By using a generative model of Petri Net, the initial value of *f* at the start of sequence replay is a value of one. Behavioral Appropriateness (ABA) is a metric that obtained by an exploration of the state space of a Petri Net and by comparing the different type of follows and precedes relationships that occur in the state space with the different types of follows and precedes relationships that occur in the event log. It defined as the proportion of number of follows and precedes relationships that the Petri Net has in common with the event log vis-a-vis the number of relationships allowed by the Petri Net. Degree Model Flexibility (DMF), as a part of ABA, is a metric that allow the flexibility in the model according to the activity relations derived from the log. It equals to 0 for a model that only allows for on particular sequence of steps, 1 for the "flower" model allowing for arbitrary execution of the contained steps.

Experiment result using real log with data information of container (in Table 1) is shown in Table 2. By applying constraints from 0-30, it shows the result of *f*-measure (fitness measure), ABA and DMF value. The result of *f*-measure and DMF are slightly decreasing when the constraints are larger. Since *f*-measure is a sequence replay of mined model from event log, it is certain that it goes lower when the constraint is larger. In the perspective of DMF, it shows that larger constraints draw a slight increment on the measure with constraint equals to 10 and 15. Afterward, it shows a better model to represent the flexibility of sequences. ABA measure shows that the behavior of the mined model is gradually increasing when the constraint is larger. It means that the process model has high proportion of precedence and follows relatioship with event log.

**Table 2.** Experiment Result from real log (full log) for proximity miner

| Number of constraints | 0 | 5 | 10 | 15 | 20 | 25 | 30 |
|---|---|---|---|---|---|---|---|
| *f*-measure | 1 | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 |
| ABA | 0.61 | 0.63 | 0.63 | 0.64 | 0.68 | 0.69 | 0.69 |
| DMF | 0.62 | 0.57 | 0.7 | 0.71 | 0.67 | 0.69 | 0.56 |
| Time (sec) | 16.615 | 16.442 | 14.533 | 14.583 | 15.584 | 15.367 | 16.175 |

**Table 3.** Comparison of experiment result from real logs (Container Data, as mentioned in Table 2) with other mining techniques (AM = alpha miner, HM = heuristic miner, PM = proximity miner)

| *f* | | | ABA | | | DMF | | | Time (sec) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| AM | HM | PM | AM | HM | PM | AM | HM | PM | AM | HM | PM |
| 0.71 | 0.98 | **0.99** | 0.72 | **0.82** | 0.69 | 0.32 | 0.47 | **0.56** | **2.961** | 4.792 | 16.175 |

## 5 Related Work

Over the last decade, there are various techniques and algorithm to mine process model from logs [2][3][4][5][9]. Those methods demonstrate the construction of process models from logs (i.e. process discovery), or the identification of discrepancies between logs and a given predefined process model (i.e., conformance testing). A process model can be compared with the process knowledge of experts by utilizing the conformance checking algorithm, which is available as one of a kind of process mining function [6]. Overall, both process discovery and conformance checking, as important components of process mining, help a process manager to enhance the productivity of business process from the point of view of process models extracted from logs.

In the domain of process mining, there are a few prior works which include prior knowledge setting. Using the traditional techniques in process mining, several approaches, i.e. alpha mining [4], region-based mining [9], allow user's intervention to the result of mined models to improve the quality of the process model. Since they use manual post processing, their modification technique toward the user requirement is not compatible and somehow unsound, e.g. modification of events that occur in sequence to be a parallel behavior (some activities may not be connected), events that occur in sequence may not represent as adjacent in the process model (some activities may not reach finish properly) etc. can result connectedness problem. Current approaches, artificial generated negative events (AGNEs) [12] and process discovery via precedence constraints [11] have an attempt to cover the limitation on traditional techniques. AGNEs has detailed description of generating artificial negative events rather generating constraint. In addition, the generated negated events, which also include log's noise, may possibly lead to false conclusions. The work on precedence constraints seems more similar to this proposed approach. However, there is no discussion on soundness property, which is important in the domain of process discovery i.e. expert give the desired constraints and lead to discover unsound process. Moreover, some other issues such as additional constraints (i.e designated start, designated end), which is very critical issue for process discovery, and interface for the convenience of users are not discussed in detail. This study attempt to cover the limitation of previous works by including user's knowledge on mining the model.

ILP Miner [3] had already been proposed to discover the process model based on the optimization problem using linear programming technique. However, it focused on the extraction of process models where all possible orderings of events must be satisfied under certain constraints. Proximity miner with user's domain knowledge aims at representing process models with additional logs behavior which only experts who can acknowledge it. Proximity miner also uses ILP and shows a method of reflecting user’s knowledge. This method is expected to reduce the time taken to discover a model compared with that of using constraints satisfaction problem [18].

## 6 Conclusion

This study proposed a new mining algorithm, called as proximity miner. It has advantages on some aspects. First, it can resolve soundness problems when it acknowledges extra behavior, which is not extracted using the current mining techniques. The extra behavior, which is invoked by users, may cause soundness problem if those behaviors do not exist in the logs i.e. the existing of parallel activities which result either clustered process model or not-connected activity. Second, it is able to detect traceable event to be a potential direct succession (adjacent) by using the proximity score. It was demonstrated that it is a useful mining tool when user intend to apply their subjective knowledge for retrieving process model.

There are open issues for further work. Semantic detection based on certain event type may show formalize model that useful for users, i.e. a schedule event which is not directly followed by complete event can be considered as a parallel activity. In addition, the existing of loop behavior may cause a clustered process model which is regarded as unsound. Finally, it also requires a method to reduce the computational time. Those remaining issues are considered as our future work.

**Acknowledgment.** This work was partly supported by the SW R&D program of MKE/KEIT [10045047].

## References

1. Kawalek, P., Kueng, P.: The Usefulness of Process Models: A lifecycle Description of how Process Models are used in Modern Organisations. In: Proceedings of the Second CAiSE/IFIPS. 1 International Workshop on Evaluation of Modelling Methods in Systems Analysis and Design, pp. 1–12 (1997)
2. Weijters, A.J.M.M., van der Aalst, W.M.P., de Medeiros, A.K.A.: Process mining with heuristics miner algorithm. Eindhoven University of Technology, Eindhoven (2006)
3. van der Werf, J.M.E.M., van Dongen, B.F., Hurkens, C.A.J., Serebrenik, A.: Process Discovery Using Integer Linear Programming. In: van Hee, K.M., Valk, R. (eds.) PETRI NETS 2008. LNCS, vol. 5062, pp. 368–387. Springer, Heidelberg (2008)
4. van der Aalst, W.M.P., Weijters, A.J.M.M., Maruster, L.: Workflow Mining: Discovering Process Models from Event Logs. IEEE Transactions on Knowledge and Data Engineering 16(9), 1128–1142 (2004)
5. van der Aalst, W.M.P.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer, Berlin (2011)
6. Rozinat, A., de Medeiros, A.K.A., Gunther, C.W., Weijters, A.J.M.M., van der Aalst, W.M.P.: Towards an evaluation framework for process mining algorithm. Eindhoven University of Technology, Eindhoven (2007)
7. Koren, Y., North, S.C., Volinsky, C.: Measuring and Extracting Proximity Graphs in Networks. Journal ACM Transactions on Knowledge Discovery from Data (TKDD) 1(3) (2007)
8. Gunther, H.O., Kim, K.H.: Container Terminals and Automated Transport Systems: Logistics Control Issues and Quantitative Decision Support. Springer, Berlin (2005)

9. Bergenthum, R., Desel, J., Lorenz, R., Mauser, S.: Process Mining based on regions of languages. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 375–383. Springer, Heidelberg (2007)
10. van Dongen, B.F., van der Aalst, W.M.P.: A Meta Model for Process Mining Data. In: Casto, J., Teniente, E. (eds.) Proceedings of the CAiSE 2005 Workshops (EMOI-INTEROP Workshop), Porto, Portugal, pp. 309–320 (2005)
11. Greco, G., Guzzo, A., Pontieri, L.: Process Discovery via Precedence Constraints. In: ECAI (2011)
12. Goedertier, S., Martens, D., Vanthienen, J., Baesens, B.: Robust Process Discovery with Artificial Negative Events. Journal of Machine Learning Research 10, 1305–1340 (2009)
13. de Amo, S., Furtado, D.A.: First-order temporal pattern mining with regular expression constraints. Data & Knowledge Engineering 62, 401–420 (2007)
14. Guns, T., Nijssen, S., De Raedt, L.: Itemset mining: A constraint programming perspective. Artificial Intelligence 175, 1951–1983 (2001)
15. Barba, I., Weber, B., Valle, C.D., Jimenez-Ramirez, A.: User recommendations for the optimized execution of business processes. Data & Knowledge Engineering 86, 61–84 (2013)
16. Rashidi, H., Tsang, E.P.K.: Novel Constraints satisfaction models for optimization problems in container terminals. Applied Mathematical Modelling 37, 3601–3634 (2013)
17. Džeroski, S.: Inductive Databases and Constraint-Based Data Mining. In: Jäschke, R. (ed.) ICFCA 2011. LNCS, vol. 6628, pp. 1–17. Springer, Heidelberg (2011)
18. Salvagnin, D.: Constraint Programming Techniques for Mixed Integer Linear Programming, Padova (2008)
19. Yahya, B.N., Bae, H., Bae, J., Liu, L.: Tool Support for Process Modeling using Proximity Score Measurement. International Journal of Innovative Computing, Information and Control 8(7B), 5381–5399 (2012)
20. van der Aalst, W.M.P., Schonenberg, M.H., Song, M.: Time Based Prediction on Process Mining. Information Systems 36(2), 450–475 (2011)

# A Methodological Evaluation of Business Process Compliance Management Frameworks

Mustafa Hashmi[1,2] and Guido Governatori[1,2]

[1] NICTA* Queensland Research Laboratory, 2 George St. Brisbane Australia
{mustafa.hashmi,guido.governatori}@nicta.com.au
[2] Queensland University of Technology (QUT) Brisbane, Australia

**Abstract.** Existing compliance management frameworks (CMFs) offer a multitude of compliance management functionalities for the modeling of specific norms for specific domain and compliance checking of normative requirements. This makes difficult for enterprises to decide on a framework suitable for their compliance requirements. Making a decision on the suitability requires a deep understanding of the functionalities of a framework. Gaining such an understanding is a difficult task which, in turn, requires specialised tools and methodologies for evaluation. Current compliance research lacks such tools and methodologies for evaluating CMFs. This paper reports a methodological evaluation of existing CMFs based on pre-defined evaluation criteria. Our evaluation highlights what existing CMFs can offer, and what they cannot. Also, it underpins various open questions and discusses the challenges in this direction.

**Keywords:** Business Processes, Compliance management, Compliance management frameworks.

## 1 Introduction

The demand for reporting compliance puts pressure on enterprises to streamline their processes within the defined limits for better transparency and effective control over their operations. Essentially, compliance is an enterprise's ability to meet all the governing regulations enforced on its business operations. This demand has become even stronger after the fall of big corporate names like Enron, American International Group (AIG) which resulted in, due to non-adherence to regulations, the emergence of regulatory acts e.g., Sarbanex-Oxley Act, BASEL-III etc. These acts place restrictions and provide guidelines for enterprises on how to perform their business operations to stay compliant, and impose severe financial and criminal penalties otherwise.

Business processes provide enterprises an abstract view of the state of the affairs on how they are achieving business objectives, and implement regulatory policies governing their business operations. That is why enterprises use business processes to verify the effectiveness of regulatory laws and policy controls. Currently, enterprises employ a number of business process compliance checking strategies: *modeling-time* where the

* NICTA is funded by the Australian Government as represented by the Department of Broadband, Communication and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-319-02922-1_10
M. Song, M.T. Wynn, and J. Liu (Eds.): AP-BPM 2013, LNBIP 159, pp. 106–115, 2013.

analysts verify the non-compliant behavior before a process can be implemented, while processes are continuously monitored at *execution-time*, and logs are audited *after-execution* for any non-compliant patterns, see [13] for more details.

To support these strategies, a large body of compliance management frameworks (CMFs) has emerged, see [3] for a list of existing approaches. Each of these frameworks bears specific functional and operational capabilities, supports specific compliance requirements and domains, and claims to be a complete compliance solution. Such a multitude creates confusion for enterprises to decide on the suitability, and accordingly generate their compliance requirements. Deciding on the suitability requires a very careful and deep understanding on various features of a CMF which, in turns, is a difficult task and requires specific tools and methodologies. So far no accepted methodologies and tools exist that can be used to evaluate various features of CMFs. We address this issue and report a methodological evaluation of the selected CMFs on how they achieve compliance. The specific questions of this paper are: *how compliance is secured*, and *whether all types of norms are supported*. Addressing these questions will provide better understanding on the various functionalities of CMFs, also it identifies issues related to the normative requirements modeling and shortcoming of existing CMFs. Therefore, a diverse community may be benefited from this evaluation to obviate the deficiencies this evaluation may highlight.

The rest of the paper is structured as follows: Section 2 describes our research approach and a detailed discussion on the normative requirements followed by a comprehensive evaluation of the selected CMFs in Section 3. A short discussion on the conducted evaluations is presented in Section 4. Section 5 discusses the related work in the problem domain and gives some pointers on the future work to conclude the paper.

## 2 Research Approach

In this section, we present our research approach to conduct this evaluation. We adopted a systematic case study based shallow evaluation strategy which allowed us to start the evaluation with minimal information available on the CMFs. We followed a three steps structured approach where, we first defined the evaluation objectives and criteria, and then selected frameworks based on pre-defined criteria:

**Evaluation Objectives:** Our objective is to examine the conceptual foundations of existing CMFs. We specifically look at the conceptual approach a framework proposes to secure compliance, and the support for the normative requirements: more specifically what constructs are provided for modeling the norms. In addition, how the norms are linked to business processes for compliance checking.

**Evaluation Criteria:** We determined a three steps selection criteria to identify representative frameworks for this evaluation (1) *level of compliance management:* this criterion describes the level of support a framework provides. We only selected CMFs which provide full compliance management support and did not consider those merely provides a compliance checking algorithm or a modeling language, (2) *requirements modeling:* this criterion allows examining how frameworks model different types of compliance requirements, and using which formal logic. Essentially, this criterion is used to identify the modeling constructs for a specific

obligation type proposed in a framework. For this purpose, we provide a classification of normative requirements which has been obtained in a systematic and exhaustive way by considering the aspect of validity of obligations or prohibitions and what it means to violate them and what happens after they have been violated. (3) *requirements linking:* this criterion allows identifying how different frameworks link the compliance requirements with business process models for compliance checking.

**Sample Frameworks Collection:** Although we reviewed and analysed several CMFs, we abstained from doing a systematic literature survey as in [6] rather we selected frameworks based on expert discussions, and mostly cited in literature. In addition, we also considered the evaluation criteria while selecting the evaluated frameworks. We believe the selected frameworks are best suited for our evaluation according to the aforementioned criteria.

### 2.1 Normative Requirements

In the legal context, norms are meant to control the behaviour of their subjects and define whether something is legal or not. Typically norms describe their application conditions and the legal effects they produce when applied. [9] provides a comprehensive list of normative effects. From a compliance perspective the normative effects of interest are the deontic effects (a.k.a normative positions). The basic deontic effects are: *obligation, prohibition,* and *permission*[1]. Consider the definitions of the basic deontic effects as defined by the OASIS LegalRuleML working group[2].

**Obligation:** A situation, an act, or actions to which a bearer is legally bound, and if it is not achieved or performed results in a violation.

**Prohibition:** A situation, an act, or actions which a bearer should avoid, and if it is achieved or performed results in a violation.

**Permission:** Something is permitted if the obligations or prohibitions to the contrary do not hold.

Obligations and prohibitions act as constraints restricting the operations of business processes. What makes them different from other types of constraints is that they can be violated, but a violation does not mean inconsistency within the process with the consequent termination of or impossibility to continue the process. Also, it is common that violations can be compensated for, and processes with compensated violations are still compliant [13]. For example, usually contracts contain compensatory clauses specifying penalties and other sanctions to counter the violation of an obligatory clause [10]. However, not all violations are compensable, and uncompensated violations mean that a process is not compliant. Permissions are a special case of deontic effects and cannot be violated. Hence, they have no explicit role in compliance; rather they can be used to define that there are no obligations or prohibitions to the contrary, or to derive other

[1] There are other deontic effects, but these can be derived from the basic ones, see [19].

[2] The OASIS LegalRuleML glossary is available at:
`http://www.oasis-open.org/apps/org/workgroup/legalruleml/download.php/48435/Glossary.doc`

deontic effects. Legal reasoning and legal theory typically assume a strong relationship between obligations and prohibitions as they are dual of each other: the *prohibition* of $A$ is the obligation of $\neg A$, or if $A$ is *obligatory* then $\neg A$ is prohibited [19].

Compliance aims at verifying whether a business process fulfills a set of obligations or not. For such a verification the first step is to determine whether and when an obligation is in force. Hence, an understanding on the lifetime of an obligation and related implications on the activities in a process is particularly important. This raises the question how long does an obligation hold for, and based on this which are the different conditions to fulfill the obligation. A norm can specify that an obligation is in force at a particular point of time, or more often indicates when it enters into force. An obligation remains active until it is terminated or removed. Accordingly, we will speak of a *punctual obligation* in the first case, and *persistent obligation* in the second.

In case of persistent obligations we can ask if we have to obey the obligation conditions for all instants in the interval in which it is in force, *maintenance obligations*, or whether meeting those conditions at least once is enough, *achievement obligations*. Furthermore, in case of an achievement obligation whether the obligation condition can be met even before the obligation is actually in force. If this is the case, then the obligation is *preemptive*, otherwise we have a *non-preemptive obligation*.

Termination of obligations is an important aspect of norms. Norms can specify the interval in which an obligation is in force. As was previously discussed that the violation of obligation conditions is what differentiates obligations from other constraints. The question is how the violation effects the obligation? Does the violation terminates the obligation or we still need to meet the conditions of a violated obligation? If we do –the obligation persists even after a violation –we speak of a *perdurant obligation*, and if not –then we have a *non-perdurant obligation*.

From the discussion above it is clear that different types of obligations have different compliance requirements. Hence, a '*one size fits all*' approach is far from being satisfactory from a conceptual point of view, and also a CMF not representing the various nuances of obligation is not conceptually sound, and it might not be suitable to provide any certification of compliance acceptable to accredited certifying organisations.

## 3 Evaluation of Frameworks

### 3.1 PENELOPE

PENELOPE (*Process Entailment from Elicitation of Obligations and Permissions* [8]) is a declarative language that captures obligation and permission constraints imposed on business processes by business policies. Aiming at providing a design-time compliance verification, the language uses an algorithm that progressively generates the *state space* and *control-flow* of a business process. The state space in a PENELOPE generated process is the set of obligations and permissions that are active at a particular state. The interaction between the generated process models flows from state to state, and all the states are enumerated until no obligation or permission holds at a state or if there is a violation which cannot be repaired. Once all the states are computed, the algorithm draws the BPMN model for a role involved in the business interaction. The tasks of the process are drawn whenever an obligation set contains all obligations fulfilled by a role

in the activity. PENELOPE allows for the modeling of interaction between all involved partners and any violations from a third partner are represented as time out events in the generated BPMN model. In addition, errors and end events are drawn if there is a violation of an obligation or permission by a role in a state. With the designed compliant process models various inconsistencies e.g. deontic conflicts can be identified.

The deontic assignments in the PENELOPE are modeled using event-calculus that provides a rich semantics to reason about the normative requirements. However, currently PENELOPE can only support achievement obligations and permissions while no other obligations types are explicitly supported as shown in Table 1. PENELOPE can model achievement obligations because they permit to explicitly define deadlines in the form of precedence rules. Prohibitions are not considered under close-world assumption (CWA) to avoid the anomalies that might occur because of incomplete knowledge about all the parties involved in the business interaction.

Violations in PENELOPE can only occur in the form of deadlock situations or temporal conflicts. Deontic conflicts cannot occur in PENELOPE generated BPMN models because the framework does not consider prohibitions or waived obligations. Moreover, no support for compensation obligation is provided because PENELOPE does not offer any mechanism to handle violations and this is left to the process modelers.

### 3.2 Process Compliance Language (PCL)

PCL (Process Compliance Language) [11] is a formal framework based on defeasible and deontic logic. It provides a conceptually rich formal foundation to model norms, and is able to efficiently capture the intuition of almost all types of normative requirements. Norms are modeled in the form of PCL rules for which the framework provides rich semantics. The state variables and the tasks in a process are represented by a set of propositional literals. $\neg$ negation, $\otimes$ non-boolean connective operator (for violations chains), and deontic operators representing obligations and permissions are used to construct the logic formulas called *PCL specifications*. The tasks in business processes are annotated with PCL specifications that are either provided by the domain experts or automatically extracted from the schemas of the databases or data sources linked to the processes [14]. These annotations are used to analyse whether the behavior of an execution path is consistent with the annotated specifications. For this purpose, a three-step algorithm is used in which first the process graph is traversed to find the set of effects for all tasks. These effects are then used to determine the norms in force for the tasks. The effects of the tasks and pertaining obligations are then compared in the last step to find any divergent behavior. The compliance of the norms is reported as full, partial, or not compliant by the algorithm.

The rich combination of defeasible and deontic logic allows PCL to model almost all types of obligations as depicted in Table 1 and other aspects of normative reasoning. This is because the use of two logics where deontic logic provides the support to model obligation's violations and chains of reparation, while the issue of partial information and inconsistent prescription is handled by defeasible logic [12]. To model the fundamental obligations PCL provides three modeling constructs: punctual ($O^p$), maintenance ($O^m$), and achievement ($O^a$), and an operator to express the orthogonal distinction between the classes of achievement obligations e.g., *premeptive obligations*.

Violations and obligations arising from the violations are major concerns in CMFs; PCL provides effective management of the violations and their compensations. For this purpose PCL defines a special contrary-to-duty non-boolean $\otimes$ connective that is used to create reparation chains for handling multiple violations of obligations. As discussed in Section 2.1 some obligations may perdure and remain in force no matter how many times they may have been violated, but currently PCL cannot handle such obligations.

### 3.3 DECLARE

Declare [5] is a prominent framework for *run-time* verification of constraint-based declarative models. A declarative model describes *what a model does* by specifying the business constraints as rules that *should not be violated*. The business knowledge in Declare is defined in terms of constraints using ConDec (Constraint Declarative [18][3]), a language which provides graphical notations to model the flows of business interactions. Declare models (also templates) are enacted by a workflow engine that is used to verify the compliant interaction between the tasks in the model. The framework includes two types of constraints i.e., *mandatory* and *optional* constraints on the process models. In a Declare model, a process instance can only be active when there is not violation of the mandatory constraints and all the constraints are fully satisfied at the end of the execution of the instance. The verification results of each constraint of an active instance are expressed as *satisfied*, *temporarily violated*, and *violated*. In case all the constraints are satisfied the activities are not executed any further, but if there is a violation state no possible further execution would be allowed to satisfy the constraints. Accordingly, in the temporarily violated state the constraints are not satisfied, but there would be a possibility to satisfy the constraints.

Business constraints (norms) in the Declare framework are modeled by means of Declare expressions which are grouped as existence, relations, choice and negative constraints. The majority of these constraints are used to express obligations while the negative constraints express prohibitions. These constraints correspond to LTL expressions that provide the semantics to the Declare graphical notations. Currently, only achievement obligations and prohibitions can be modeled in the Declare Model, while no other norms types can be explicitly modeled, see Table 1. Since achievement obligation defines deadlines and the obligation condition must be true for at least once, the support for such obligations is only available because the tasks in the Declare model with such constraints will be performed in some future time. However, persistence and preemptiveness of obligations cannot be expressed. Constraints expressing maintenance obligation, on the other hand, can be problematic in Declare because the obligation conditions must hold in all instances throughout the execution of the process. There might be some situations when the applicable maintenance obligation constraints might not be present thus there will be deadlock in the course of interaction between the tasks. Declare is able to identify conflicts among constraints in the model; it does not provide any support to handle violations because of the lack of the declarative nature of the LTL and the non-deterministic behavior of the process models. Hence, in case of a violation the

[3] From Nov 2012, the name of ConDec language has been changed to Declare see `http://www.win.tue.nl/declare/2011/11/declare-renaming/`

interaction between the tasks in the Declare model will be stopped and no further activity can be performed. Accordingly, it is not possible to express permissions and so are compensations and perdurant obligations.

### 3.4 BPMN-Q

BPMN-Q (Business Process Modeling Notation-Query [1]) is a query based automated compliance checking framework capable to answer YES/NO type answers to query questions. The framework can model control-flow, data flow and conditional flow related compliance rules using visual patterns. These visual patterns are translated into LTL formulas for checking the structural compliance of a processes model. The framework adopts a systematic approach to generate the patterns of compliance rules in the form of query templates. These templates are used to identify the set of process models subject to compliance checking in the process repository. Compliance checking is carried out in several steps. First, BPMN-Q sub-graphs are extracted from the process repository using temporal query templates. The query processor only extracts processes that structurally match the query template. These sub-graphs are then reduced by eliminating irrelevant activities and gateways, and translated into a Petri Net model to generate the state space. Alongside the state space generation, BPMN-Q queries are translated into LTL formulas which are then fed into a model checker together with the generated state space. In turn, the model checker yields YES/NO to indicate whether the extracted process models comply with the query templates.

The framework uses a visual language BPMN-Q to express various types of compliance rules. The language provides visual notations similar to the standard BPMN notations. These visual notations provide the constructs to model compliance rules. Currently, the framework is able to handle almost the same types of obligations as the Declare framework with the exception of maintenance obligations (cf. Table 1). Maintenance obligations are expressed using the *global space presence* pattern which enables the execution of an activity that is required in all process instances.

In BPMN-Q no conceptual or formal constructs have been provided that can expressively model permissions. Whereas prohibitions are represented by global space absence to prevent the execution of some activities. Unlike the Declare framework, BPMN-Q is able to handle violations for which a violation handling approach has been discussed in [2]. Finally, compensations and perdurant obligations are not supported because of the use of LTL as underlying formalism to model compliance rules.

### 3.5 SEAFLOWS

SeaFlows [17] is a CMF for the verification of semantics constraints. It incorporates a graphical language providing primitives to capture process related complex business rules. These compliance rules are modeled in the form of *first-order logic* predicates equivalent and instantiable to compliance rules graphs (CRG). SeaFlow employs a structural compliance checking strategy for the verification of compliance rules where node relations are verified against the imposed constraints. The verification is done in three steps: in the first step a set of structural templates based on the queries on the relations of nodes in the process models is automatically derived. Then, the process

model is checked against the derived templates to detect any non-compliant structural templates. The queried templates are then aggregated and fed into the SeaFlows compliance module for further compliance report in the last step. The compliance results are based on the execution of traces of the process models where a process model is fully compliant when all the activities in the trace comply with the instantiated rule. Whereas a No is returned to indicate rule violations when no activity in the execution trace satisfies the rules.

To model compliance rules, the SeaFlows framework adopts a compositional graph-based modeling formalism allowing for the modeling of the typical antecedent-consequencet structure of rules. These graphs serve as placeholder for the first order logic representation of the relevant rules. Although SeaFlows is able to model achievement obligations which stipulate the occurrence of some event in future time by means of occurrence pattern, the framework is not able to capture other modalities e.g., punctual, maintenance, permissions, and compensation as depicted in Table 1. Moreover, compensations and perdurant obligations arising from the violation of the primary obligations cannot be modeled because first-order logic is not suitable to reason about the normative requirements [16].

**Table 1.** Normative Requirements Support in Existing CMFs

| Framework | Obligations | | | | | Permissions | Prohibitions | Violations |
|---|---|---|---|---|---|---|---|---|
| | Punctual | Achievement | Maintenance | Compensation | Perdurant | | | |
| PENELOPE | – | + | – | – | – | + | – | – |
| PCL | + | + | + | + | – | + | + | + |
| DECLARE | – | + | – | – | – | – | + | – |
| BPMN-Q | – | + | + | – | – | – | + | + |
| SEAFLOWS | – | + | – | – | – | – | + | + |

## 4 Discussion

We have evaluated five CMFs using a set of pre-defined evaluation criteria. This methodological evaluation examined the salient features of CMFs, and *what is lacking* in terms of technical support in compliance domain especially for the modeling of normative requirements. The evaluation results are shown in Table 1, where the available support for modeling norms is indicated with '+', not supported with '−'.

It is clearly evident that only a fraction of normative requirements are supported as none of the evaluated CMFs is capable of supporting all types of norms. For example, PENELOPE is only able to support obligations and permissions. It is unable to model other obligation modalities, and violations because Event Calculus is not suitable for reasoning of legal constraints. Contrary to that, PCL supports almost all types of norms because of the non-monotonic characteristics of the formal logic it uses. The combination of defeasible and deontic logic allows PCL to provide reasoning for deontic modalities and violations especially for temporally varying obligations, e.g., achievement obligations and their persistence over time. As can be seen, PCL does not support perdurance obligation. DECLARE and BPMN-Q are LTL based frameworks, and only address 'structural compliance' where the tasks are defined by the constraint models. These frameworks cannot capture the intuition of all types of obligations, violations, and their compensations. DECLARE can only support achievement obligations and

prohibitions while BPMN-Q can support achievement, maintenance, and prohibitions only. Generally it is highly desirable that a formal language is expressive enough to cover most of the properties and properties of the environment of the unit under verification. In addition, it should also support the complex properties from simpler ones, but temporal logic lacks such support because it has no conceptual relative correspondence to the legal domain, thus cannot expressively model the properties of the norms. The SeaFlows framework, on the other hand, can only support achievement, prohibitions and violations while other obligations modalities cannot be modeled. Because first order logic is not suitable to reasoning about the normative requirements as it does not provide operators to represent the properties of norms.

## 5 Conclusions

We presented a methodological evaluation of some existing CMFs using a shallow, but sound methodology where we examined the conceptual foundations under pre-defined evaluation criteria. Specifically we looked at the conceptual approaches existing CMFs use to deal with the normative requirements related to the regulatory compliance.

[3] offers a literature survey based on the generalisability and applicability of business process compliance frameworks. Their evaluation is based on the reported implementation results from the surveyed frameworks, while [6] compares the functional and non-functional capabilities of regulatory compliance management (RCM) solutions from a BPM perspective using a large set of evaluation criteria. Similarly, [4] studies various frameworks using a four point criteria including the study of modeling languages that are used to model business processes and rules. [7] conducted a comparative analysis of formal frameworks used to model the compliance rules, and [20] investigates the practices of regulation analysis and the approaches aiming to achieve and maintain regulatory compliance of given regulation from an Information Systems and services perspective. Our evaluation is complementary and different from these studies because we primarily evaluated existing CMFs to examine what they can do in terms of providing round-up compliance, and what constructs they provide to model different types of normative requirements. In addition, we also examined whether existing CMFs can provide reasoning support for all types of norms or not.

Our evaluation results portray somewhat a *bleak picture* when it comes to see how existing frameworks represent the legal knowledge for compliance checking because none is able to support all types of norms. Primarily this is because of the formal language each framework uses to model the norms. This highlights an exigent need for new compliance rules modeling languages with sound theoretical and formal foundations to effectively model and faithfully represent the legal knowledge, thus would increase the effectiveness of CMFs. We plan to conduct the formal semantics evaluation of these frameworks where we will examine the modeling behavior and constructs provided, and their correspondence to a specific modeling language. For this purpose, we have designed a formal framework in which we have provided detailed ontology and formal semantics for each of the normative requirements described in Section 2.1, [15]. The planned evaluation will allow us to examine the expressive power of the compliance rules modeling languages, and to identify what is required in this direction?

## References


1. Awad, A., Decker, G., Weske, M.: Efficient Compliance Checking Using BPMN-Q and Temporal Logic. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240, pp. 326–341. Springer, Heidelberg (2008)
2. Awad, A., Weske, M.: Visualization of Compliance Violation in Business Process Models. In: Rinderle-Ma, S., Sadiq, S., Leymann, F. (eds.) BPM 2009. LNBIP, vol. 43, pp. 182–193. Springer, Heidelberg (2010)
3. Becker, J., Delfmann, P., Eggert, M., Schwittay, S.: Generalizability and Applicability of Model-Based Business Process Compliance-Checking Approaches – A State-of-the-Art Analysis and Research Roadmap. BuR - Business Research Journal 5(2), 221–247 (2012)
4. Cabanillas, C., Resinas, M., Ruiz-Cortes, A.: Hints on how to face business process compliance. In: III Taller de Procesos de Negocio e Ingenieria de Servicios PNIS 2010 in JISBD 2010, vol. 4, pp. 26–32 (2010)
5. DECLARE. Declarative process models, `http://www.win.tue.nl/declare/`
6. El Kharbili, M.: Business process regulatory compliance management solution frameworks: A comparative evaluation. In: APCCM 2012, CRPIT, vol. 130, pp. 23–32 (2012)
7. Elgammal, A., Turetken, O., van den Heuvel, W.-J., Papazoglou, M.: On the formal specification of regulatory compliance: a comparative analysis. In: Maximilien, E.M., Rossi, G., Yuan, S.-T., Ludwig, H., Fantinato, M. (eds.) ICSOC 2010. LNCS, vol. 6568, pp. 27–38. Springer, Heidelberg (2011)
8. Goedertier, S., Vanthienen, J.: Designing Compliant Business Processes with Obligations and Permissions. In: Eder, J., Dustdar, S. (eds.) BPM Workshops 2006. LNCS, vol. 4103, pp. 5–14. Springer, Heidelberg (2006)
9. Gordon, T.F., Governatori, G., Rotolo, A.: Rules and norms: Requirements for rule interchange languages in the legal domain. In: Governatori, G., Hall, J., Paschke, A. (eds.) RuleML 2009. LNCS, vol. 5858, pp. 282–296. Springer, Heidelberg (2009)
10. Governatori, G.: Representing Business Contracts in RuleML. International Journal of Cooperative Information Systems 14(2-3), 181–216 (2005)
11. Governatori, G., Rotolo, A.: A Conceptually Rich Model of Business Process Compliance. In: Proceedings of APCCM 2010, vol. 110, pp. 3–12. ACS, Inc., Australia (2010)
12. Governatori, G., Rotolo, A.: Norm compliance in business process modeling. In: Dean, M., Hall, J., Rotolo, A., Tabet, S. (eds.) RuleML 2010. LNCS, vol. 6403, pp. 194–209. Springer, Heidelberg (2010)
13. Governatori, G., Sadiq, S.: The Journey to Business Process Compliance. In: Handbook of Research on Business Process Management, pp. 426–454. IGI Global (2009)
14. Hashmi, M., Governatori, G., Wynn, M.T.: Business Process Data Compliance. In: Bikakis, A., Giurca, A. (eds.) RuleML 2012. LNCS, vol. 7438, pp. 32–46. Springer, Heidelberg (2012)
15. Hashmi, M., Governatori, G., Wynn, M.T.: Normative Requirements for Buisness Process Compliance. Technical report, Queensland University of Technology, Brisbane, Australia (2013)
16. Herrestad, H.: Norms and formalization. In: ICAIL 1991, pp. 175–184. ACM (1991)
17. Ly, L.T., Rinderle-Ma, S., Dadam, P.: Design and verification of instantiable compliance rule graphs in process-aware information systems. In: Pernici, B. (ed.) CAiSE 2010. LNCS, vol. 6051, pp. 9–23. Springer, Heidelberg (2010)
18. Pesic, M., van der Aalst, W.: A declarative approach for flexible business processes management. In: Eder, J., Dustdar, S. (eds.) BPM Workshops 2006. LNCS, vol. 4103, pp. 169–180. Springer, Heidelberg (2006)
19. Sartor, G.: Legal Reasoning: A Cognitive Approach to the Law. Springer (2005)
20. Turki, S., Bjekovic-Obradovic, M.: Compliance in e-government service engineering: State-of-the-art. In: Morin, J.-H., Ralyté, J., Snene, M. (eds.) IESS 2010. LNBIP, vol. 53, pp. 270–275. Springer, Heidelberg (2010)

# Improvement of Patient Safety in u-Hospital: A Pattern-Based Approach for Handling Patients' Abnormal Situations

Junho Moon and Dongsoo Kim*

Department of Industrial and Information Systems Engineering,
Soongsil University 369 Sangdo-Ro, Dongjak-Gu, Seoul, Korea
{jmoon,dskim}@ssu.ac.kr

**Abstract.** This paper presents a pattern-based approach for handling abnormal situations of a patient appropriately. In order to identify and model patterns of abnormal situations, we apply ontology technology. Several important patterns have been identified and they can be managed by the proposed system. We have designed a system architecture of integrating BPM and ontology, and developed core part of the proposed system. The integrated system is called u-PMS (ubiquitous Patient Management System). It can detect meaningful events and abnormal situations which affect health status of a patient, and manage healthcare processes to cope with the abnormal situations in process centric way. By applying context awareness technology and ontology technology to healthcare sector, patient safety can be improved significantly.

**Keywords:** Context Awareness, Patient Management System (PMS), u-Hospital, Ontology, BPM.

## 1 Introduction

The information technology has been applied to the medical environment in order to enhance patients' safety and service quality. Recently, Researches on Patient Management System (PMS) using context awareness technology have been studied. Various and heterogeneous data are dispersed in a variety of different systems. This kind of data plays only a trigger role in PMS [2], [3], [8]. However, it is necessary to organize and to filter the dispersed data in order to build more efficient systems in hospital.

In this paper, we present an advanced patient safety management system by applying context awareness, ontology, and BPM technologies. We have identified and modeled several abnormal situation patterns based on ontology. The proposed system can detect the occurrence of abnormal situations using various ubiquitous devices such as RFID and smart sensors, and then manage healthcare business processes for handling such abnormal situations precisely and promptly through the integrated BPM system.

* Corresponding author.

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-319-02922-1_10

M. Song, M.T. Wynn, and J. Liu (Eds.): AP-BPM 2013, LNBIP 159, pp. 116–120, 2013.

# 2 Backgrounds

## 2.1 Necessity and Effectiveness of BPM in Medical Environment

The BPM system plays a role in facilitating healthcare business processes and removing unnecessary work in medical environment. It can visualize administrative and clinical processes and automate such processes. We can expect substantial benefits when the BPM system is applied to the medical environment. It makes the smart response to the environmental change and reduces the costs of IT projects.

The unnecessary medical work procedure can be eliminated in medical service by separating and explicitly managing the business processes from hospital information systems. The medical malpractices or errors can be reduced through the work process visualization because it makes the process confusion lower. The process confusion has been pointed out to be an important factor of the medical malpractices. It is necessary to adopt the BPM system to the medical environment for the integrated medical service management from the point of overall process [1], [4], [5].

## 2.2 Necessity of Data Management Based on Ontology

Ontology in medicine integrates the dispersed data and filters the meaningful things among these. This makes the precise and prompt response to the patient's abnormal situations based on the data. Technological convergence is no longer new, and the medical environment has already adopted various information technologies. As a result, the amount of medical data is becoming huge and a solution to use this data efficiently is in urgent. Ontology might be one solution for those needs in medicine [6], [7], [9].

# 3 System Architecture

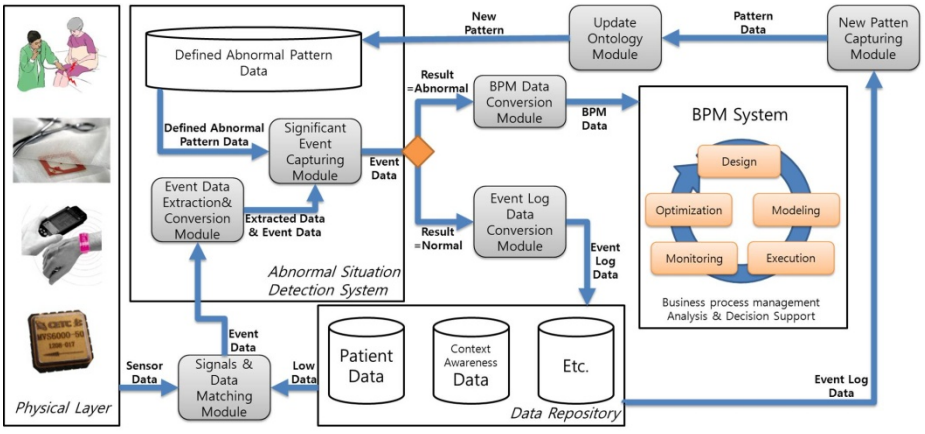



**Fig. 1.** System Architecture

Figure 1 indicates the architecture of the proposed u-PMS (ubiquitous Patient Management System). In this system, the data is collected in medical environment using context awareness technology. It is stored in database and patients' abnormal situations are recognized based on the ontology. The recognized abnormal situations are reported to the BPM system and the business process, which is defined in BPM, offers how to handle abnormal situations.

This system integrates the heterogeneous data from context awareness tools and extracts valid events immediately to handle patients' abnormal situations. This procedure is specified through the link with BPM system and prevents mistakes in the process as emphasized in the previous section. Continuous improvement of the business process perspective is also the advantage of using BPM system.

# 4 Implementation and Example Scenario

## 4.1 Detection of Abnormal Situations

Abnormal situation detection system in Figure 1 performs two important functions. Firstly, it defines patients' abnormal situations as patterns based on ontology as shown in Figure 2. The defined abnormal situation is also defined as a schema type using the OWL (Web Ontology Language). Secondly, the system changes the data recognized by the context awareness tools into OWL schema type. Then, the abnormal situation can be identified by the comparison of OWL schema type with the defined abnormal situation patterns. Only meaningful events are transferred to the BPM system.

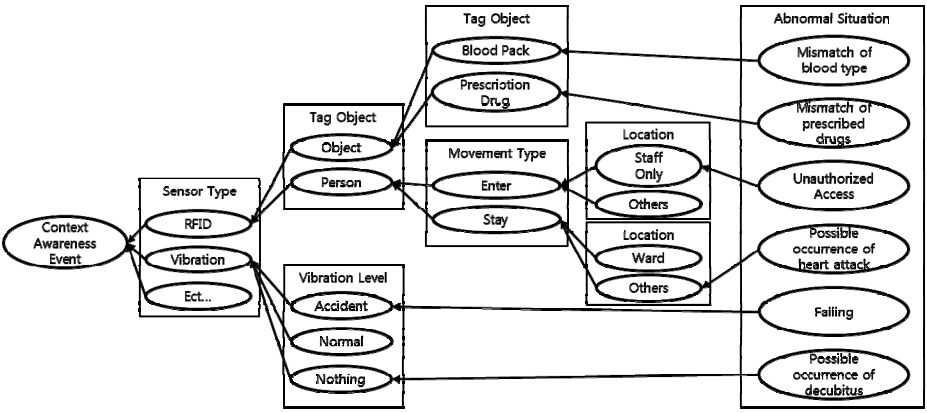



**Fig. 2.** Conceptual model for modeling patient's abnormal situation patterns based on ontology

## 4.2 Integration with BPM System

The BPM system processes the events from the abnormal situation detection system according to the defined business process. We use an open source BPM solution

named uEngine in order to execute processes for handling detected abnormal situations. In this stage, when an abnormal situation is detected, a corresponding process can be started by the status of database or external signals using 'conditional start' or 'signal start' in BPMN standard specification.

### 4.3 Process Flow of Example Scenario

This section describes a scenario of an abnormal situation and explains the process flow of the scenario. Let's consider a hospitalized patient. If the patient falls from the bed, the vibration sensor attached to the patient can sense the occurrence of falling. The sensor ID and measured value from the sensor are transferred to 'Signals & Data Matching Module'. This module generates 'Event Data' by referencing the basic information and context awareness information from the 'Data Repository'. Patient name, disease, status, location, sensor ID and so on are stored in 'Event Data'. 'Event Data' is transferred from 'Signals & Data Matching Module' to 'Event Data Extraction & Conversion Module'.

'Event Data Extraction & Conversion Module' extracts a partial data for comparison with defined abnormal situation patterns and converts it to the form of ontology schema in Figure 3. 'Extracted Data' and 'Event Data' are transferred to 'Significant Event Capturing Module'. This module checks whether the event is an abnormal situation or not by comparing 'Extracted Data' with 'Defined Abnormal Pattern Data'. According to the result of the comparison, 'Event Data' is transferred to the next module. If the situation is normal, 'Event Data' is sent to 'Event Log Data Conversion Module'. 'Event Data' is transformed into the form of 'Event Log Data' and stored in 'Data Repository'. If the situation is abnormal, 'Event Data' is sent to 'BPM Data Conversion Module'. 'Event Data' is transformed into the form of 'BPM Data' in order to invoke the business process for handling the abnormal situation and transferred to the 'BPM System'.

```
<owl:Class rdf:ID="EventSchema">
  <rdfs:subClassOf rdf:resource="#VibrationLevel">
    <owl:hasValue rdf:resource="#accident" />
    <rdfs:subClassOf rdf:resource="#SensorType">
      <owl:hasValue rdf:resource="#Vibration" />
      <rdfs:subClassOf rdf:resource="#ContextAwarenessEvent" />
    </rdfs:subClassOf>
  </rdfs:subClassOf>
</owl>
```

**Fig. 3.** Ontology schema of extracted event data

## 5 Conclusions

We proposed ubiquitous Patient Management System (u-PMS) based on BPM and ontology technologies. Some existing studies have also applied the context awareness technology to the medical environment and used BPM or ontology. However, the contribution of this paper is the integration of each information technology for immediate and accurate management. The context awareness technology allows the system procedure to be smart. The strengths of BPM system such as agility, visibility, flexibility and efficiency, and even the advantages from ontology (integrity and expendability) can be incorporated in this new system.

However, limitations of this paper can be summarized as follows. Since the process in the medical environment is highly complex and changeable, the establishment of BPM will be much demanding. And also there are so many synonyms in medical terminology, therefore defining a comprehensive medical ontology is quite challenging.

**Acknowledgements.** This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2010-0020943).

## References

1. Basu, A., Kumar, A.: Research Commentary: Workflow Management System in e-Business. Information System Research 13(1), 1–14 (2002)
2. Choi, J.S., Kim, D.: Design and implementation of Wireless Patient Monitoring System for postoperative patients on general care floor. Informatics for Health and Social Care (Posted online on May 9, 2012)
3. Fuhrer, P., Guinard, D.: Building a Smart Hospital using RFID technologies. In: Proceedings of the 1st European Conference on e-Health, Fribourg (2006)
4. Hieb, B.R.: BPM/Workflow Boosts the Value of Computer-Based Patient Record System. Gartner (2005)
5. Holingsworth, D.: Workflow Management Coalition: The Workflow Reference Model, WfMC, No. TC00-1003 (1995)
6. Kataria, P., Juric, R., Paurobally, S., Madani, K.: Implementation of Ontology for Intelligent Hospital Wards. In: Hawaii International Conference on System Sciences, p. 253 (2008)
7. Lasierra, N., Alesanco, A., García, J.: An ontology approach to manage individual patient profiles in home-based telemonitoring scenarios. Information Technology and Applications in Biomedicine (2010)
8. Aspden, P., Corrigan, J.M., Wolcott, J., Erickson, S.M.: Patient safety: achieving a new standard for care (2004)
9. Kiong, Y.C., Palaniappan, S., Yahaya, N.A.: Health ontology system. In: Information Technology in Asia, pp. 1–4 (2011)

# Erratum to: Asia Pacific Business Process Management

Minseok Song[1], Moe Thandar Wynn[2], and Jianxun Liu[3]

[1] School of Technolgy Management,
Ulsan National Institute of Science and Technology, Ulsan, South Korea
msong@unist.ac.kr
[2] Information Systems School, Queensland University of Technology,
Brisbane, OLD, Australia
m.wynn@qut.edu.au
[3] School of Computer Science and Engineering,
Hunan University of Science and Technology, Xiangtan, China
ljx529@gmail.com

## Erratum to:
## M. Song, M.T. Wynn, and J. Liu (Eds.)
## Asia Pacific Business Process Management
## DOI: 10.1007/978-3-319-02922-1

The book was inadvertently published with an incorrect name of the copyright holder. The name of the copyright holder for this book is: © Springer-Verlag Berlin Heidelberg. The book has been updated with the changes.

The updated original online version for this book can be found at
DOI: 10.1007/978-3-319-02922-1

M. Song, M.T. Wynn, and J. Liu (Eds.): AP-BPM 2013, LNBIP 159, p. E1, 2013.

# Author Index