International Journal of Latest Engineering and Management Research (IJLEMR)
ISSN: 2455-4847
www.ijlemr.com || Volume 08 – Issue 02 || February 2023 || PP. 58-75

Discovering Complex Processes from Event Data

MOUAFO MAPIKOU Gaelle Laetitia
Faculty of Sciences
Universty of Yaounde 1

ATSA ETOUNDI Roger
Faculty of Science
Universty of Yaounde 1

ABESSOLO ALO'O Ghislain
Faculty of Science
Universty of Yaounde 1

Abstract: The omnipresence of using automated management tools like Enterprise Resource Planning (ERP) systems to support business processes has enabled recording a large amount of event data (stored in event logs) which indicates the behavior of these processes. These data can be used to audit the information system in order to discover and understand the real management that is carried out in the daily execution of business processes, and this is the scope of process discovery. Traditional process discovery techniques capture the behavior of an event log in a business process generally represented as a petri net. However, given the rapid increase of event data, it becomes difficult to discover understandable processes using process discovery algorithms, if so, at the risk of producing complex processes that ultimately are not understandable by end-users. Empirical studies have shown that large event data originally derive from complex processes for which representation in the form of petri nets is not suitable. This paper suggests to discover a process represented as proclets rather than petri nets. Proclets are an extension of petri nets designed for complex process modeling, and represented as a set of sub-processes interrelated. The paper presents a framework that enables the discovery of proclets. The divide and conquer approach is used to allow obtaining a structure in sub- processes. The proclets thus discovered are proved to improve understandability according to their structure. A case study is carried out to show the effectiveness of our findings.

Index Terms: Event Data, Process Discovery, Complex pro- cess, Petri nets, Proclets

1. Introduction

Nowadays, information systems are taking a promi- nent place in the execution of business processes they support. Examples of well-known and widely used industrial information systems are Customer Relationship Management (CRM) systems and ERPs (van Beijsterveld & Van Groenen- daal 2016) (Enterprise Resource Management) systems which manage nearly anything that can happen within an organization related to finance, resource management, sales management, supply chain management, etc. These systems have enabled the recording of a large amount of event data describing the behaviour of real executed business processes within the organizations. Typically, each action performed by a user (e.g. print sales invoice) is recorded in the system as an event. Events contain different information generally organized in four perspectives (Tiwari et al. 2008, Van Der Aalst et al. 2011), (1)-control-flow perspective (the activity performed by the user), (2)-resource-perspective (the human resource or the role who executed the activity), (3)- data perspective (data manipulated to execute the activity), and (4)-time-perspective (timestamp of the activity). Analyzing such event data has become a major topic in knowledge discovery approaches such as process discovery.

Process discovery is an emerging event data analysis tech- nique that has proved its usefulness by providing a new insight into business processes. Traditional process discovery tech- niques (Janssenswillen et al. 2017) capture the behaviour of an event log (file containing event data related multiple execution of a business process) in a process model. The discovered process model can be used audit the information system in order to understand the real management that is carried out in the daily execution of business processes. Typically, process discovery enables understanding what is the real process that people follow within the organization; where do people deviated from the expected execution and also, what are the bottlenecks in the process. This shows the practical importance of process discovery beyond the scientific ones. So, for the sake to continuously propose process models useful to business users, an important practical quality metric is understandability (Reijers & Mendling 2011, Mapikou & Etoundi 2016) used to ensure that the discovered process model can be easily interpretable and analyzable by beneficiaries.

To illustrate the importance of understandability of discov- ered business processes, let us consider a well-known data set related to the oncological process of a hospital in Amsterdam, available on the repository http://data.3tu.nl/repository/. Imag- ine that after receiving a patient in the oncological gynecology, there was a medical error in its follow-up which led to its dead. Investigations are carried out to understand the causes of the dead and at which level of the oncological process the medical error occurred. Process discovery algorithms are suitable for this investigation. Figure 1 presents the pro- cess discovered by the alpha-algorithm (Van der Aalst et al. 2004) on this data set, using the ProM 6.5 toolkit (available at www.processmining.org); this process

represents the real process thatis followed up in practice for the gynecological oncology, but it is of no effect for business users who want to use it for investigations because it can't be interpreted; where to start in this process? Where to end? What are the paths of the process? Why are some activities disconnected from the graph? Why is the process divided into two parts? In summary, how to analyze this process? This process is not understandable since it is large and it is not possible to clearly identify its paths, activities, arcs and others.

Researchers have been trying to explain the non- understandability of some processes obtained by process dis- covery. As a result, investigations reveal that for modern and mature organizations as it is mostly the case nowadays, there are two aspects to consider. On the one hand, business processes are originally complex (too many activities, arcs and split and join constructs for concurrency and synchronization), and petri nets are no more suitable for their representation (van der Aalst et al. 2009) while this model is the fundamental model used by most of process discovery algorithms. On the other hand, since business processes are complex, their execution generates a large amount of data that become diffi- cultly handled by process discovery algorithms (Leemans et al. 2015). Indeed, discovered process are generally represented as petri nets or specialized notations (Van Der Aalst 2011) tailored towards process mining like causal nets, heuristic nets and process trees which do not allow understandable representation of complex processes. However, in the litera- ture, a specific notation called proclets (an extension of Petri nets represented as a set of sub-processes interrelated) has been defined for complex process representation, but according to our known, existing process discovery algorithms are not yet able to discover such a process model. We claim that discovering proclets rather than petri nets will improve the understandability of the discovered process, and therefore produce more insights to business users.

This paper proposes a framework that uses a divide and conquer approach to discover proclets from event data. This framework takes an event log as input and proceeds in three stages: decomposition, discovery and interconnection. The decomposition stage aims to split input data into subset of data. Two log decomposition approaches based on the control flow perspective have been proposed in the literature (Van der Aalst 2013), to distribute the cases or activities of an event log over sub-logs. In this work, we propose a novel decomposition approach for which in addition to the control flow perspective, also considers the resource perspective and uses it as the splitting criteria. For the discovery stage, an algorithm is presented to discover sub-processes from sub-logs obtained at the decomposition stage. The objective of the interconnection stage is to interrelate sub-processes obtained at the discovery stage, in order to build a unique coherent system called proclet system. Some management rules are defined for this purpose. A case study on the process shown in figure 1 is carried out to show the effectiveness of our findings.

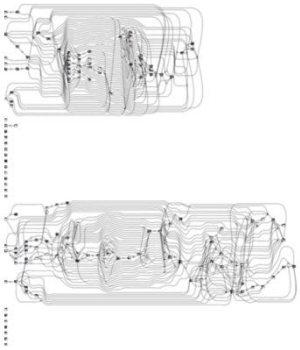

Fig. 1: Non-understandable business process

The remainder of this paper is organized as follows: section 2 presents a literature review of related work on process discovery; section 3 presents some preliminary concepts useful to understand our proposal. These concepts concern process modelling techniques like petri nets and proclets, event data and an algorithm for control flow discovery. Section 4 presents our proposal;. Section 5 presents the case study; Section 6 concludes the paper and provides some directions for future research.

2. Related Work

Process discovery is a knowledge extraction approach that extracts a business process from event logs. This section summarizes existing works on process discovery. To under- stand this summary and related concepts, this section first of all summarizes related work on process modelling before presenting process discovery algorithms.

A. Process Modelling

The literature review on process modelling presents var- ious business process models, ranging from semi- formal business process models (graphical representation, without formal semantic) to formal business process models (graphical representation and formal semantic).

An important categorization has been done on existing business process models. This categorization focusses on the perspective (four perspectives that are control flow, data, resource and timestamp perspectives) that is taken into account in the modelling and proposes two main categories:

- Process/Activity-centric : is the main focus of most of the modelling languages and process management systems. This category is centered on the control flow perspective; it is explicitly specified the activities to be performed and the execution relationship between them. The information and data flows are hidden in the model and are very often unknown to the process. This category comprises semi-formal notations (like Business Process Modelling Notation(Chinosi & Trombetta 2012), Event driven Process Chains, Yet Another Workflow Chains), (Scheer 2012) and formal notations (transition systems, Petri nets Van der Aalst (1998), Murata (1989)
- 2) Artifact-centric : aims at conceptually integrating the process and data layers in the business logic; the mod- elling of a business process combines control flow and data perspetives. Business processes are built in terms of collections of artifacts that encapsulate data and have an associated lifecycle (Bhattacharya et al. 2007). This category comprises two notations that are proclets (van der Aalst et al. 2009) and guarded attribute grammars (Frost 1993).

Two metrics have been defined to evaluate the quality of a process model: understandability and soundness.

- Understandability- A process model should be as simple as possible, and easy to understand. This dimension puts emphasis on the one hand on simpler models which are not overly complex, e.g., they are not extremely large and the density of arcs is low (Mendling, Neumann & Van Der Aalst 2007);and on the other hand, the ease of interpretation and cognitive capabilities;
- Soundness - A process model should be *sound* i.e. free of deadlocks and livelocks. This dimension allows guaranteeing that a process will always terminate for any path followed and there is no dead state.

B. **Process Discovery**

In the past fifteen years, many process discovery tech- niques have been proposed. One of the major drivers for process miners is the development of the ProM framework (Verbeek et al. 2010). This supporting framework developed at TU/Eindhoven allows the development of all kind of process intelligence techniques. One of its most important features is the underlying file format for process event logs, formerly known as MXML, but recently improved to XES (eXtensible Event Stream).

The most relevant of these works can be classified into three approaches (De Weerdt et al. 2012): deterministic approaches: (1)-deterministic algorithms produce a defined and reproducible result: given an algorithm, this family of process discovery approaches extracts some footprint that is used to directly construct a process model; (2)-Divide and conquer approaches: try to break the problem into sub-problems. Rather than scanning the event-log in a single pass, the event log is recursively split into sub-logs and each sub-log is scanned separately; (3)-Computational Intelligence approaches: use an evolutionary approach, i.e., the log is not directly converted into a model but uses an iterative procedure to mimic the process of natural evaluation. Based on these approaches, several algorithms have been proposed.

Van Der Aalst in 2003 (Van der Aalst et al. 2004) proposed the alpha-algorithm, foundational to understand the process mining field and its challenges. The author proves that it can learn an important class of workflow nets, called structured workflow nets, from complete event logs. The alpha-algorithm assumes event logs to be complete with respect to all allowable binary sequences and assumes that the event log does not contain any noise. Therefore, this algorithm is sensitive to noise and incompleteness of event logs. Moreover, the original alpha-algorithm is incapable of discovering short loops or non- local, nonfree choice constructs.

Alves de Medeiros et al. in (Alves de Medeiros et al. 2004) improved the alpha-algorithm to mine short loops and named it alpha+. Other extensions of the original alpha-algorithm have been proposed. Wen et al. (Wen et al. 2007) proposed the alpha++ that is able to detect non-free choice constructs. Furthermore, Wen et al. (Wen et al. 2009) have also proposed to take advantage of both start and completed event types in order to detect concurrency and named it the alpha#-algorithm.

Alves et Medeiros (de Medeiros et al. 2007) proposed the genetic algorithm in order to deal with non-trivial constructs like non-free choice, invisible tasks and duplicate tasks. The main motivation was to benefit from global searches, allowing detecting non-local patternsand ensuring a fair robutness. Genetic Miner defines its search space in terms of causal matrices. These matrices express task dependencies only, yet they are closely related to Petri nets.

In order to deal with the robustness problem of the alpha- algorithm, Weijters et al. (Weijters et al. 2006) developed HeuristicsMiner. In particular, HeuristicsMiner extends the original alpha-algorithm in that it applies frequency infor- mation to discover three types of relationships between ac- tivities in an event log: direct dependency, concurrency and not directly-connectedness. HeuristicsMiner can discover short loops and non-local dependencies but is not able to detect duplicate activities.

Gunther and van der Aalst (Gu¨nther & Van Der Aalst 2007) proposed Fuzzy Miner, an adaptive simplification and visualization technique based on significance and correlation measures to visualize the behavior in event logs at various levels of abstraction. The particularity of Fuzzy Miner is that it can also be applied to less structured, or unstructured processes of which the event logs cannot easily be summarized in concise, structured process models. Despite the fact that this approach is an interesting data exploration technique, a Fuzzy model cannot be translated to a formal Petri net which severely limits a comparative evaluation to other process discovery techniques.

Ferreira et al.in (Ferreira & Ferreira 2006) proposed Integer Linear Programming (ILP) to discover a process model. The discovered process model is represented in terms of the case data preconditions and effects of its activities. The main con- tribution of this work is the integration of the business process management lifecycle of process generation, execution, re- planning and learning. Later, Lamma et al. (Lamma et al. 2007, Van der Werf et al. 2008) describes the use of ILP to process mining, assuming the presence of negative sequences to conduct the mining algorithm.

Goedertier et al. (Goedertier et al. 2009) proposed the algo- rithm called AGNEsMiner that represents the discovery task as first-order classification learning on event logs, supplemented with artificially generated negative events. Like Genetic Miner, AGNEsMiner is capable of constructing Petri net models from event logs and has been implemented as a mining plugin in the ProM Framework. The main contribution of this algorithm is the ability to learn the discriminating conditions that determine whether an event can take place or not, given a history of events of other activities.

Greco et al. (Greco et al. 2006) proposed an algorithm based on machine learning, called DWS mining. This algorithm extends previous discovery algorithms by putting forward an abstraction method that aims to produce a taxonomy of process models and allows analyzing the behavior contained in the event log at different levels of detail. This idea has also been implemented in Petrify Miner by Van Der Aalst (van der Aalst et al. 2006, Van der Aalst et al. 2010).

Aalst et al. proposed the petrify Miner (Van der Aalst et al. 2010), which is a two-step approach to find a trade-off between precise and general process. In the first step, a transition system should be constructed from the traces in an event log. In the second step, this transition system is synthesized by means of theory of regions in such a way that a Petri net can be constructed. The advantage of this algorithm is that it allows controlling generalization. However, it cannot deal with noisy data. Genet Algorithm proposed by Carmona et al. (Carmona et al. 2010) is similar to Petrify Miner and also allows the construction of a Petri net from a transition system.

Leeamans et al. (Leemans et al. 2013*b*,*a*, 2014*a*) proposed a set of algorithms called Inductive Miners (IM) to deal with infrequent behavior in the log and unsound process models. These algorithms construct the directly-follows graph from an event log. IM is currently one of the leading process discovery technique due to its flexibility, formal guarantees and scalability; however it still has problems to generate understandable process models from large event data.

The table 1 below summarizes the most relevant process discovery models on reasonable amount of data (let us say medium and simple logs) in terms of approach used, resulting process model, strengths and weaknesses. Strengths and weak- nesses in this comparison are evaluated on different levels of granularity. Among others, we have the ability to deal with noisy/ incomplete data, short loops, concurrency, and unstructured processes; the ability to handle split and join constructs and frequency of events; the ability to discover sound/fitting process models, the computation time.

Later, scalable process discovery has been introduced by Redlich et al. (Redlich, Gilani, Molka, Drobek, Rashid & Blair 2014). In the paper, authors presented a model for the concept of scalable dynamic process discovery that is considered as an adapted process discovery application of Event-driven Business Process Management, independent of any additional model input. Evaluation is based on event granularities (activ- ity level events vs. process level events) or perspective support (whether resource, trace or other information provided or not). However, the data perspective provided by events is not taken into account. (Redlich, Molka, Gilani, Blair & Rashid 2014*b*) is a refinement of (Redlich, Gilani, Molka, Drobek, Rashid & Blair 2014) with the use of the Construct Competitor Miner (CCM) algorithm (Buijs et al. 2012) that enables the discovery of run-time process discovery from event streams.

Evermann et al. in (Evermann 2016), proposed a model that uses the map-reduce technique to discover processes from scalable distributed event data. A set of mappers and reducers are defined to compute log-based ordering relations from distributed data. The authors presented parallelizable implementations of this model on

log-based algorithms like alpha-algorithms and frequency-based algorithms like Heuris- tic Miner (HM) (Weijters et al. 2006) and Flexible Heuristic Miner (FHM) (Weijters & Ribeiro 2011) algorithms. Using Amazon Reduce Service (ARS), experiments proved the scal- ability of the proposed model which is able to handle more than millions of event traces. Though this model simplifies the mining process, the overall mining time complexity remains unchanged.

Leemans et al. in (Leemans et al. 2015, 2016), dis- cussed the applicability of other discovery algorithms such as CCM(Redlich, Molka, Gilani, Blair & Rashid 2014*a*), Evolu- tionary Tree Miner (ETM)(Bergenthum et al. 2009), Integer Linear Programming (ILP)(Van der Werf et al. 2008), and commercial tools with no executable semantics like Fluxicon Disco (FD)(Gu¨nther & Rozinat 2012) and Perceptive Process Mining (PM)(Leemans et al. 2014*b*) in highly scalable envi- ronments. Afterwards, they proposed a model for process dis- covery from the event logs of tens of thousands of events and tens of hundreds of activities. Rather than recursively splitting on the log as the basic Inductive Miner (IM), authors present a model implementing a refined version of the IM called the Inductive-Miner-Directly-Follows based (IMD)(Leemans et al. 2015) that recursively split the directly follows graph until a base case is encountered. This model ensures quality criteria such as fitness and produces sound models. Moreover, it is introduced the single pass quality criteria which refers to the ability to discover a process model by scanning the log only once. Directly-follows graphs are proved to be computed in a single pass on large and complex data and can be paral- lelized though its computation time remains high. In (Leemans et al. 2016), the computation time of the IMD on noisy and incomplete event logs of 100000000 traces and processes of 10000 activities are considerably reduced. Evaluation metrics such as model-model and model-log comparisons that apply the divide-and-conquer strategy to evaluate the fitness and the precision of discovered processes have been used. Although this model has been proved to be highly scalable and aligned with most of process discovery properties, resulting discovered processes lack of ease to understand abilities.

As a summary on scalable process discovery algorithms, IM and alpha-algorithms are the most aligned with process discovery properties. Both guarantee neither fitting models nor understandability. IM guarantee soundness in multiple passes over the log though understandability is not guaranteed. FHM guarantees neither fitness, nor soundness or understandability though it has been proven to highly scalable trough its single pass property. In summary, none of the scalable process discovery is able to gurantee understanability, which poses a real problem because this property is as important as other properties for researchers in general and especially for end- users of the organization.

3. Research Methodology

The strong need of extracting relevant information from a data heap containing data related to business process exe- cutions demands analysis techniques. This is where process discovery steps in. It helps researchers and data scientists in extracting useful insights from the data deluge related to process execution and available in information systems. Traditional process discovery models need to be adapted for keeping handling understandability in process discovery from data deriving from several execution of complex pocesses. The objective of this paper is to adapt one of the existing process discovery algorithms in order to allow handling large data of complex processes and improve understandability of discovered process.

Management systems like ERP systems support the op- eration of an entire organization which has large processes. This allows recording various information related to business process executions. Moreover, as different executions of the same process may have different behaviours and manipulate different data, according to the respective needs, it is almost difficult to describe the behavior of a system using only one process definition. These issues have led to imagine another way of representing processes as a set of interrelated processes, hence the proclets. Proclets are a collection of Petri nets communicating through messages. Building such a process model implies providing answers to the following questions:

- how to decompose an entire business process into sub- processes?
- how to define the communication semantic between sub- processes? i.e. how to define messages exchanged between sub-processes? and how to define communication channels?

TABLE I: Comparison of Process Discovery Models

Approach	Model	Output Process	Strenghts	Weaknesses
Deterministic	alpha	Petri net	very simple, isomorphic rediscoverability	Short loops, Noisy and incomplete data, split and joins constructs, event frequency
Deterministic	alpha+, alpha++, alpha#	Petri net	short loops, soundness	Noisy and incomplete data, split and joins constructs, event frequency
Deterministic	Petrify Miner	Transition system	fitness	State explosion problem, Unsound models
Computational Intelligence	Genetic Miner	Causal net	noise, incomplete data	High computation time (crossover and mutation)
Computational Intelligence	Duplicate Genetic Miner	Causal net	noise + incomplete data + duplicate activities	High computation time
Computational Intelligence	AGNEsMiner	Causal net	noise + event discrimination	High computation time
Divide and Conquer	Heuristic Miner	Heuristic net	noise, infrequent behavior	Unsound and non-fitting models
Divide and Conquer	Flexible Heuristic Miner	Heuristic net	noise, infrequent behavior	Unsound and non-fitting models
Computational Intelligence	Fuzzy Miner	Fuzzy net	Unstructured process	No executable semantics, Unsound and non-fitting models
Computational Intelligence	Integer Linear Programming	Petri net	Fitness	Unsound model
Computational Intelligence	Evolutionnary Tree Miner	Process tree	Soudness	High response time
Divide and conquer	Inductive Miner	Process Tree	Infrequent, incomplete data, soundness	Unfitting model
Divide and conquer	Construct Competitor Miner	BP-domain constructs	Soundness	Non-fitting model
Computational Intelligence	Maximal Pattern Mining	Petri net	Soudness and fitness	Duplicated activities

We propose to use divide and conquer approaches. Our idea is to break large datasets into subsets of data easily handled by existing techniques. For this, we propose a formal framework based that used the decomposition approach to split data, before applying a discover algorithm. In order to address the understandability problem, we suggest to discover a particular type of process model that we called Proclets. Proclets dis- covered in this work are simply a set of sub-processes inter- connected, each sub-process being a Petri net. This process model is more suitable for complex process modeling. We claim that it will be more interpretable than others, thus can be a good candidate to improve understandability.

The assessment of our framework is a case study on the real data set used in the motivating example (figure 1). The aim of this case is to practically show how the understandability of the process obtained by our framework has been improved. Moreover, since that every process modelling recommends to prove that the resulting process is free of deadlocks and it terminates, the soundness of the discovered proclet is theoretically established.

4. Basic Concepts

This section presents some concepts necessary to understand this work. This concerns event data, process models such as Petri-nets, workflow-nets, and the collaborative process model used in this work.

A. Event data

Event data refers to data contained in event logs. An event log is basically a table. It contains all recorded events that relate to executed business activities. Each event is mapped to a case. A process model is an abstraction of the real world ex- ecution of a business process. A single execution of a business process is called a process instance. They are represented in the event log as a set of events that are mapped to the same case. The sequence of recorded events in a case is called a trace. The model that describes the execution of a single process instance is called a process instance model. A process model abstracts from the single behavior of process instances and provides a model that represents the behavior of all instances that belong to the same process. Cases and events are characterized by classifiers and attributes. Classifiers ensure the distinction of cases and events by mapping unique names to each case and event. Attributes store additional information that can be used for analysis purposes. These informations concern the activity that have been executed (control flow

perspective), the role or the function that performed the activity (resource perspective), data used to perform the activtiy (data perspective) and the time at which the activity was performed (timestamp).

B. Petri-nets

Petri-nets are one of the oldest technique (Van der Aalst 1998) for specifying business processes. Originally, petri-nets consist of places, transitions and directed arcs connecting places and transitions. Connections between two nodes of the same type are not allowed. Places are represented by circles and transitions by rectangles. Once the network structure has been established, tokens move from one place to another, thereby forming a transition. Since transitions may change the network status, i.e. the number and the position of tokens, they are active components such as events or tasks; tokens represent physical objects or information objects.

It has been defined a simplified version of Petri nets for Process discovery in which a place is called an input place of a transition iff there exists a directed arc from the place to the transition. A place is called an output place of transition iff there exists a directed arc from the transition to the place. A transition is said to be enabled iff each of its input places contains at least one token. An enabled transition may fire. The firing of a transition consumes one token from each of its input places and produces one token in each of its output places. This paper considers a synthetic version of Petri nets called place-transition Petri nets.

Formally, a place-transiton Petri net is defined as follows:
Petri-net: a Petri-net N is a tuple (P,T,F) where:

- *P is a finite set of places,*
- *TisafinitesetoftransitionssuchthatP∩T=∅,*
- *$F \subseteq (P \times T) \cup (T \times P)$ is a set of directed arcs called flows.*

C. Workflow-nets

Workflow-nets (Van der Aalst 1998) are improved petri- nets that allow easy representation of business processes by adding new concepts (the unique source and sink places). For example, Workflow nets may have a hierarchical structure, explicit split and and join nodes that enhance parallelism and exclusivity relationships between activities. Places represent conditions and tokens, and carry information on specific processes. However, data are not explicitly represented either in petri-nets or in Workflow nets: in fact, process execution is controlled solely by the control flow, and no data transport is allowed. More precisely, a Petri-net is called a workflow net if and only if (a)- it contains a source place (an input place with no input transition); (b)- it contains a sink place (a place with no output transitions) (c) - it is strongly connected, i.e., there is a directed path between any pair of nodes in the net. Formally, a WF-net is defined as follows:

Workflow-net: let N = (P,T,F) a petri-net; N is a WF-net if and only if:

- *P has an input place (called source place) such that .i = ∅ ;*
- *P has an output place (called sink place) such that i. = ∅;*
- *N is strongly connected, i.e., there is a directed path between any pair of nodes in N.*

D. Proclets

A proclet (Van Der Aalst et al. 2001, Mans et al. 2010) is an artifact-centric object-specific process modeled as a Petri net that can be seen as a workflow equipped with a knowl- edge base containing information on interactions with other proclets. Interactions between proclets are based on messages exchanged through ports and each message sent or received is stored in the knowledge base of the corresponding proclet. Proclets interact via channels. A channel is a medium used for carrying messages from a proclet to another. Via a channel, a message can be sent to a specific proclet or group of proclets (multicast). These messages are called performatives. Based on the channel properties, different types of interactions are supported, for example, push/pull, synchronous/asynchronous, and verbal/nonverbal. Proclets are connected to channels via ports. Each port has two attributes: (a)- the cardinality and (b)-the multiplicity. The cardinality indicates the number of receivers of performatives exchanged via the port. The mul- tiplicity indicates the number of performatives exchanged via the port during the lifetime of an instance of the class. The life cycle of a particular type of proclets and ports are specified in terms of proclets class. Formally, a proclet is defined as:

Proclet: a proclet P = (N, ports) consists of a Petri net N = (S,T,F) and a set of ports, where:

- *some initial transition $t \in T$ has no input place (i.e., $\{s|(final,s) \in F\} = \emptyset$);*
- *each port $p = (T^p, dir_p, card_p, mult_p)$:*

- *is associated to a set $T^P \subseteq T$ of transitions;*
- *has a direction of communication $dir_p \in \{in, out\}$ (i.e. receives or sends message respectively);*
- *a cardinality $card_p \in \{?, 1, *, +\}$ and;*
- *a multiplicity $mult_p \in \{?, 1, *, +\}$ and*

- *each transition is attached to at most one input port and to at most one output port, i.e., $\forall t \in T$ holds $|\{p \in ports | t \in T^p, dir_p = in\}| \leq 1, |\{p \in ports | t \in T^p, dir_p = out\}| \leq 1$.*

Proclet system: a proclet system is a tuple $P = (\{p1, p2, ..., pn\}, C)$ where:

- *$\{p1, p2, ..., pn\}$ is a finite set of proclets and*
- *C is the set of channels. Each channel $(p,q) \in C$ connects an output port p to an input port q, $p,q \in \bigcup_{i=1}^{Sn} ports_i$, $dir_p = out$, $dir_q = in$*

Instead of having an initial marking like Petri-nets, a proclet has an initial transition which will produce the first tokens in the net. Figure 2 shows two proclets: quote and order. Each proclet has three ports. The output port of the transition accept has cardinality + (messages are sent to multipe orders) and multiplicity 1 (only one message per quote). The input port of addCD has cardinality 1 (each input message triggers one of the addCD transitions) and multiplicity + (at least one message is received during the lifecycle of an order). The set of the two proclets quote and order together with the communication channels exchanged is called a proclet system.

E. Alpha-algorithm for Control Flow Discovery

The Alpha-algorithm discovers one possible workflow net (Van der Aalst et al. 2004) from the observed event logs under the assumption that there is no noise in data contained in the event log . The algorithm first identifies log-based ordering relationships between activities, based on the direct precedence relationship. For example, when an activity always follows an- other activity, but not vice-versa, there is a causal relationship between the activities. If an activity follows another one and vice-versa, these activities are parallel. If there is no direct precedence between two activities then there is no relation between them.

a) Log-based ordering relationships: let L an event log on a set of activities A, $a,b \in A$

- *Direct precedence: $a >_L b$ if and only if there is a trace $\sigma = (t1,t2,t3,...,tn)$ and $i \in \{1,...,n-1\}$ such that $\sigma \in L et t_i = a and t_{i+1} = b$*
- *Causal relationship: $a \rightarrow_L b$ if and only if $a >_L b$ and $b \not>_L a$*
- *Free: $a \#_L b$ if and only if $a \not>_L b$ and $b \not>_L a$*
- *Parallelism: $a \parallel_L b$ if and only if $a >_L b$ and $b >_L a$.*

The Alpha-algorithm uses these relations to discover a workflow-net from an event log. The discover process embod- ies eight steps: (1)-identify the set of transitions; (2)-identify initial transitions; (3)-identify ending transitions; (4)-build the set of places; (5)- deduct the set of maximal places from the set of places (if there is a causal relationship between two transitions, then there is a place between these transitions); (6)-build the final set of places by adding the source and sink places (build according respectively to the set of initial and ending transitions) to the set of maximal places; (7)- build the set of directed arcs ; (8)- return the workflow-net consisting in the set of places of step 6, the set of transitions of step 1 and the set of arcs of step 7. For a more detailed definition of this algorithm, the reader must refer to (Van der Aalst et al. 2004).

5. Proposal

This section presents a framework to discover a collab- orative process model from large event data containing in- formation related to multiple perspectives. According to the description of the collaborative process given in section 4.4, discover such a process model means to be able to identify sub-processes given a entire process and clearly establish how these sub-processes communicate. The proposed frame- work proceeds in three stages as illustrated in figure 2: (1)-decomposition, (2)- discovery of sub-processes and (3)- Interconnection of sub-processes.

Recall that traditional process discovery algorithms only focus on the control flow perspective of the process and produce processes that solely describe the execution order of activities/tasks. The idea in this work is to extent the representation with the others perspectives that can lead to a more realistic process and therefore improve its understand- ability. Thus, the representation of the discovered process will combine control flow,

resource and data perspectives. The control flow perspective is used to represent the execution order between activities; The resource perspective is used to represent the organizational aspects of the process; and the data is used to highlight the data flow within the process.

A. Decomposition

The aim of the decomposition step is to split the event log into reasonable sub-logs. The idea is to associate each participant (not in terms of individuals but in terms of function or role) of the process with its running queue. So, we build as many sub-logs as participants involved the initial log. Participants are identified by the resource perspective in the log. For this, we define a high-level function decompose() to decompose the event log. This function is summarized into two actions: (1)-extraction of business process elements to (2)-splitting of the log.

The first action aims to extract elements of the business process and their relationships, used for splitting and commu- nication. These elements are grouped into three sets: the set of roles (roles involved in the log), the data − to − activity set (associates each data to an activity) and the activity−to−role set (associate each activity to a role). For this, we define a high-level procedure called BPElement(). The procedure iterates over every event recorded in the event log and checks if the event has any of the required elements attached. At every step, the appropriate value sets and relations are updated. Finally, after completing the procedure, we have an in-memory business process model, i.e., business process elements with data and relationships between these elements.

The second action uses relationships between elements obtained by the procedure BPElement(), to derive a set of sub- logs. This function begins by instantiating the different sub- logs to the empty-set and after, makes a partitioning across all roles, in order to determine the different sub-processes. After, the function iterates over traces of the log, and for each trace, each role is assigned a sub-trace. The sub-trace represents the set of the activities performed by the role in the intial trace. If there is no activity performed by a role in a given trace, then the sub-trace of this role is empty. Thus, each role has its running queue, and this is considered as a sub-process and, there is as many sub-logs as roles involved in the initial log. In each sub-log, the execution order of activities in each sub- process is given by the order of the activities in the initial event log traces.

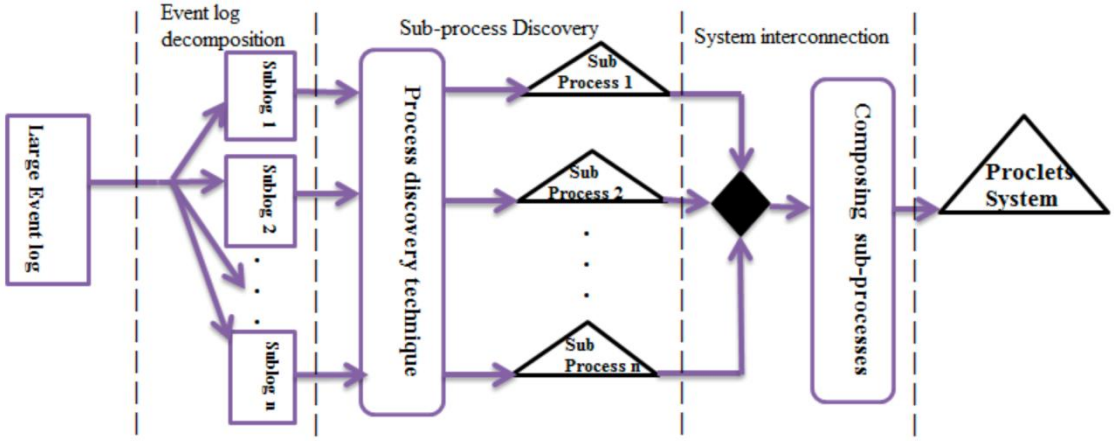

Fig. 3: Discovery Framework

TABLE II: An Event log

Activity	Role	Activity	Role
a	role 1	b	role 1
c	role 2	d	role 2
e	role 1	f	role 1
g	role 2	h	role 2
i	role 2		

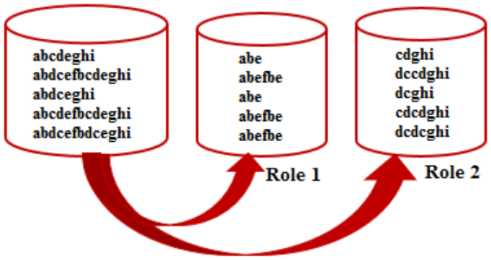

Fig. 4: An illustration of the decomposition

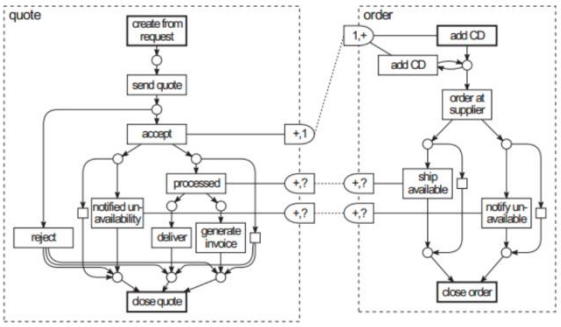

Fig. 2: The CD shop process Fahland et al. (2011) modeled as a proclet system

For example, if activity 1 and activity 2 are two activities of a sub-process, moreover, if activity 1 precedes activity 2 in a trace of the event log, then activity 1 will also precede activity 2 in the sub-process. The causal dependency relation defined in the alpha-algorithm is used here.

Table 3 and figure 3 illustrate the decomposition principle. The log initially contains nine activities (a,b,c,d,e,f,g,h,i) and two roles (role 1 and role 2) (table 3). This log has been split into two sublogs, each sublog corresponding to a role (role 1 and role 2) (see figure 4). In this case, each sublog has as many traces as traces in the initial log but the number of activities per trace of in sublogs is not equal(this may be different in other situations according to our decomposition principle).

B. Discovery

The previous step has produced a set of sub-logs corre- sponding to the set of roles involved in the initial log. The actual challenge is to associate each sub-log to a business process (sub-process). The function discovery() is defined for this purpose.

The function discovery() iterates over the set of roles. For each role, the associated sub-log is used as input of a discovery algorithm to discover the corresponding sub-process represented as a Petri net. At the end of the function discovery, the set of sub-processes is built; it remains to make them communicate.

There are some clarifications concerning the discovery algorithm. Every discovery algorithm in the literature that is able to generate a Petri net can be a candidate. Conforming with the scalable part of the related work presented in section 2.3 (establishing alpha and HM as the best candidates), , the alpha-algorithm has been chosen as the discovery algorithm. However, since the native alpha-algorithm cannot directly produce a sub-process in terms of petri net that respects the proclet specification, this algorithm has been refined to alpha R(alpha refined). Indeed, the alpha-algorithm extract a workflow net, that is a petri net with a unique input place and a unique output place; whereas the sub-process of the proclet system is a petri net that has a unique input transition and a unique output transition. The algorithm alpha R just integrates this difference.

The algorithm alpha R is quite similar to the alpha- algorithm. It takes an event log as input and generates a process in eight steps. In step 1, it is checked which activities do appear in the log to which we add two silent transitions . (silent initial transition and a silent final transition). These silent transitions represent the unique input and output tran- sitions, fondamental difference with the alpha-algorithm. The second step identifies the set of beginning transitions, i.e. first activities in traces of the log; The third step identifies the set of ending activities, i.e. last activities in traces of the log ; steps 4,5 and 6 build the set of places as done in the basic alpha-algorithm; step 7 builds the set of arcs which consists in the set of arcs obtained by the alpha-algorithm together with the two arcs used to inter-relate silent transitions to the net.

Example: let us consider the log L, L_{r1} and L_{r2} of the example 2. According to the algorithm α_R, these logs will be respectively converted into:

- L_R *= {[τI,(a1,r1,d1), (a2,r2,d2),(a3,r1,d3),(a4,r2,d4),τO],[τI,(a1,r1,d1),(a2, r2, d2), (a4, r2, d4), (a3, r1, d3), (a5, r1, d4), τO]},*
- L^R_{r1} *= {[τI,(a1,r1,d1),(a3,r1,d3),τO],[τI,(a1,r1,d1), (a3,r1,d3),(a5,r1,d4),τO]} and* L^R_{r2} *= {[τI,(a2,r2,d2), (a4,r2,d4),τO],[τI,(a2,r2,d2),(a4,r2,d2),τO]}.*

One can immediately observe that transitions τ_I and τ_O have been added respectively at the beginning and the end of each trace of each log. The corresponding footprints are given in table 3,4 and 5. These tables contain relationships obtained by the α-algorithm together with relationships created by silent transitions. τ_I is

always in causal relation with first activities *(τI →L_R a1, τI →L^R_{r1} a1, τI →L^R_{r2} a2)* and last activities are always incausal relation with τO.

C. Interconnection

The aim of this step is to express the communication seman- tic of sub-processes of the overall proclet system. Accordingto the definition of the proclet system, this communication is established through ports and channels of each sub-process.

The semantic of this communication is established on the basis of data exchanged and order between activities. Two perspectives are used for this purpose: data perspective and control flow perspective. If there is a causal relationship be- tween two activities of two different sub-processes, then both sub-processes are interrelated. Moreover, two sub-processes are also interconnected if they share one or more data (mes- sages).

The idea is that: "if there is a data d (data perspective) belonging to two activities activity1 and activity2 of two different sub-processes respectively P rocess1 and P rocess2, then there exists a message m between Process1 and Process2". In addition, if there is a causal dependency (control flow perspective) between activity and activity in the initial 12 event log, then Process1 is the sender of the message and Process2 is the recipient i.e. the flow between these two activities goes from P1 to P2. Messages (building communication channels) are used to interconnect sub-processes. Below, are management rules that summarize the functioning of the communication channels:

- Each message concerns two transitions (the receiver (in- put port) and the sender (output port)) and corresponds to a communication channel in the collaborative system;
- The input ports of a transition are the ports formed by the current transition and any other distinct transition with which there is a causal relationship;
- The output ports of a transition are the ports formed by any other distinct transition that has a causal relationship with the current transition;
- The ports of a transition are the union of its input ports and its output ports;
- the ports of a sub-process are the union of ports of its transitions;
- The set of messages represents the set of communication channels.

Formally, this section aims at constructing a proclet system *PS = (S,C)* where:

- *S = {Pr1, ..., Prn} represents the set of proclets with Pri = (αR(Lri), ports);*
- *C is the set of communication channels of the proclet system.*

$$\forall a_i \in T_L, \begin{cases} port_{out}(a_i) = (\{a_i, a_j\}, out, 1, *) | a_i \rightarrow_L a_j, \\ \qquad a_j \in T_L, \forall a_j \neq a_i (1) \\ port_{in}(a_i) = (\{a_i, a_j\}, in, 1, *) | a_j \rightarrow_L a_i, \\ \qquad a_j \in T_L, \forall a_j \neq a_i (2) \\ ports(a_i) = \{port_{in}(a_i), port_{out}(a_i)\} (3) \end{cases}$$

$$\forall P_{ri} \in S, P_{ri} = (\alpha_R(L_{ri}), \{\cup ports(a_i), a_i \in T_{L_{ri}}\}) (4)$$

$$C = \{\bigcup(p,q) | \exists P_{ri}, P_{rj} \in S, \exists a_i, a_j \in T_L \wedge a_i \in T_{L_{ri}} \wedge a_j \in T_{L_{rj}} \wedge a_i \rightarrow_L a_j \wedge p = port_{out}(a_i) \wedge q = port_{in}(a_j)\} (5).$$

$port_{in}(a_i)$ (1) and $port_{out}(a_i)$(2) represent respectively the input and output ports of the transition a_i. The output port

TABLE III: Footprint of L_R

	τ_I	a_1	a_2	a_3	a_4	a_5	τ_O
τ_I	#	→	#	#	#	#	#
a_1	←	#	→	#	#	#	#
a_2	#	←	#	→	→	#	#
a_3	#	#	←	#	‖	→	#
a_4	#	#	←	‖	#	#	→
a_5	#	#	#	←	#	#	→
τ_O	#	#	#	#	←	←	#

TABLE IV: Footprint of L^R_{r1}

	τ_I	a_1	a_3	a_5	τ_O
τ_I	#	→	#	#	#
a_1	←	#	→	#	#
a_3	#	←	#	→	→
a_5	#	#	←	#	→
τ_O	#	#	←	←	#

TABLE V: Footprint of L^R_{r2}

	τ_I	a_2	a_4	τ_O
τ_I	#	→	#	#
a_2	←	#	→	#
a_4	#	←	#	→
τ_O	#	#	←	#

concerns a_i together with all the transitions that are causally related to a_i ($a_i \rightarrow_L a_j$). Similarly, the input port concerns all the transitions that are causally related to a_i ($a_j \rightarrow_L a_i$) together with a_i. The ports of a transition is the combination of its input port and its output port (3), and always the cardinality (one receiver) 1 and the multiplicity $*$ (0 or more message instances). This definition of the ports of a transition allows to ensure that each transition has at least one input port and at least one output port, which conforms with the definition of a proclet. The ports of a proclet is the union of the ports of its transitions (4). Channels connect an output port of a proclet to an input port of another proclet(5). The union of such channels forms the channel set of the proclet system. The coherence of the proclet system is guaranteed if there is not an isolated proclet in the system.

D. Proclet system Metamodel

Fig. 4 presents the metamodel summarizing the functioning of the resulting proclet system. This metamodel is divided into two parts: the MOF metamodel and the Core-based metamodel. Recall that The OMG (Atkinson & Kuhne 2003) defined four modeling levels: (1)-M0 (real system, modeled system); (2)-M1 (model of the real system defined in a certain language); (3)-M2(metamodel defining the language) and (4)- M3 (the meta-metamodel defining the metamodel). The last level M3, also called MOF is meta-circular and can be defined by itselsves. It corresponds to the unique meta-metamodel used as the basis for the definition of all meta-metamodels. The core-based and MOF metamodels of the diagram rep- resent respectively the level 2 (M2) and the level 3 (M3). The MOF part of this diagram describes the basis of petri-nets. A Petri-net consists of transitions (TransitionConfiguration class) and places (PlaceConfiguration class). Tokens (Token class) are moving from place to place to enable transitions. The Initialization class initializes the net by distributing tokens in places. As a proclet is basically a Petri net having commu- nication ports, in the core-bsed metamodel, a proclet consists of transitions (Transition class), places (Place class) and ports (P ort class). Moreover, a proclet can be seen at different levelof granularity, hence the SubProclet class. In this case, the sub-proclet in a proclet is considered as a transition and is similar to concept of sub-process within a process in BPMN 2.0 (Chinosi & Trombetta 2012) modeling. Each proclet corresponds to a role (Role class) in the process execution and a role is assimilated to a resource (Resource class). A proclet system (ProcletSystem class) is a set of proclets communicating trough channels (Channel class).

6. Case Study

The case study is based once more on the oncology work- flow at the Academic Medical Center (AMC) in Amsterdam available at http://data.3tu.nl/repository/ .

A. ProM (Process Mining) toolkit

ProM is an extensible model that supports a wide variety of process mining techniques (?Claes & Poels 2012) related to process discovery and conformance checking in the form of plug-ins. It is platform independent as it is implemented in Java, and can be downloaded free of charge. It can be used to extract processes from business process event logs and answer the following questions: how many cases (or process instances) are in the log? How many tasks (or audit trail entries) are in the log? How many resources are in the log? Are there

running cases in the log? Which resources work on which tasks? This tool is used on each sub-log to automatically extract the associated Petri-net using the algorithm α_R.

B. Analyzing results

The BPIC11 contains 1143 traces i.e. 1143 process in- stances, 150291 events and 600 activities. According to the characterization described in section 3.1, the BPIC11 is a complex and medium event log. It involves in total 15 services considered here as roles: radiotherapy, nursing ward, obstetrics and gynecology clinic, general lab clinical chemistry, medical microbiology, internal specialism clinic, radiology, pain clinic, pharmacy laboratory, special lab genetic metabolic diseases, pathology, operating rooms, nuclear medicine, special lab radiology, anesthesiology clinic.

According to our approach, this log is split into 15 sub-logs, each one associated with a given role. Let L the original log andL_i asub-log($i \in 1..15$).Let$T$ atraceof$L$,T is inL_i if and only if there exists at least one activity in T executed by the role corresponding to L_i. This rule considerably reduces the number of traces per sub-log, therefore, reducing the extraction time in ProM. Given the number of traces in L, the complexity and the different levels of granularity of the process, the process extracted using the α-algorithm module implemented in the ProM tool is difficult to analyze. However, splitting the log L into 15 sub-logs according to the numbers of roles improves the execution time, and it can be therefore eas- ily executed by the algorithm α_R to generate 15 sub-processes (proclets). Figure 4 presents the pathology sub-process and its relationship with other sub-processes. The place represents a place of the pathology process which communicates with four other sub-processes (anesthetic, pain clinic, MDO and radi- ology) illustrated by transitions in double rectangles. Figure 5 presents a zoom on the content of the pathology process. One can observe that it has 9 transitions: 'create pathology meeting', 'prepare pathology meeting', 'pathology meeting', 'receive pathology report', 'request pathology slides', 'request additional colorings', 'request pathology slides', 'request regis- tration for pathology meeting'. The process starts by creating a pathology meeting and ends by closing the pathology meeting. The communication with sub-processes of anesthesiology, pain clinic, medical microbiology and radiology is established respectively via transitions register for anesthetic meeting, register for clinical meeting, register for MDO meeting and register for radiology meeting. This communication expresses the fact that in the event log L, there is a trace T containing 2 consecutive events, the first one corresponding to either the transition 'register for anesthetic meeting', or 'register for clinic meeting', or 'register for MDO meeting' or 'register for radiology meeting' and the second one corresponding to the transition 'prepare pathology meeting'.

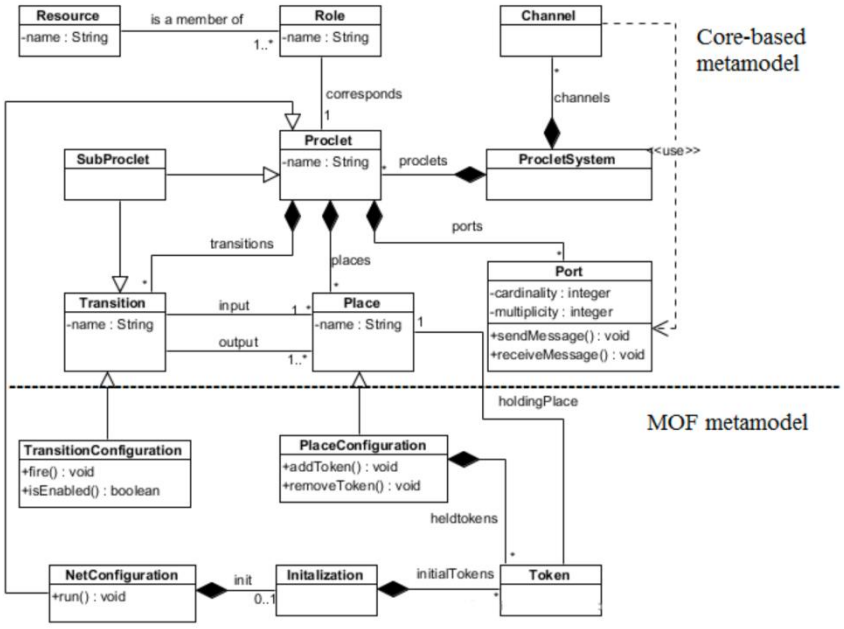

Fig. 5: Metamodel of the proclet system

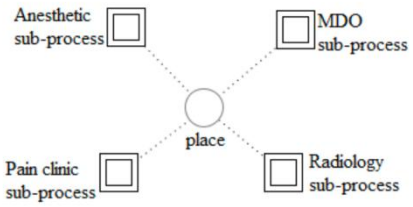

Fig. 6: Pathology sub-process and its relationship to other sub-processes

C. Understandability

This section discusses the understandability of the discovered process model. In the related work section, we adopted to define under- standability as ease of interpretation, i.e. the degree to which information contained in a process model can be easily un- derstood by a reader of that process. Based on this definition, Reijers et al. in (Reijers & Mendling 2011) identified two main categories of factors that influence the understandability: model factors (related to the process model) and personal factors (related to the reader of the process model). Both categories are theoretically discussed here, based on some established cognitive metrics.

Model factors are decomposed into four metrics: abstraction gradient, hard mental operations, hidden dependencies and secondary notations.

- Abstraction gradient is defined as the grouping capabil- ities of a process notation. Authors suggest to decom- plexify process model as much as possible in order to improve understandability. In the proposed framework, decomposing the process into sub-processes is our way of decomplexifying the process. We think that moving from a unique complex process to a set of more simple process may lead to an easier interpretation of the process since it allows more simply identifying the part of the process.
- Hard mental operations refer to the difficulty to analyse a larger process model compared to a simpler pro- cess model in the sense that calculating its reachability graph is a NP-complete problem. Authors in (Reijers & Mendling 2011, Mendling, Reijers & Cardoso 2007) suggest decomposition approaches to deal with it. This suggestion is already integrated to the proposed frame- work.
- Hidden dependencies: are defined as the interdependen- cies (split and join constructs) that are not completely visible in the process model. This dimension is illustrated on the process of figure 1. It is impossible to identify pat- terns of the process because of too many arcs connecting patterns (AND join, OR join, AND split, OR split, XOR) between activities.
- Secondary notations: refer to any extra information that is not part of the process formalism. This information includes labels, layout and others. Increasing secondary notations like labels in a process particularly a complex and large process, will improve its understandability.

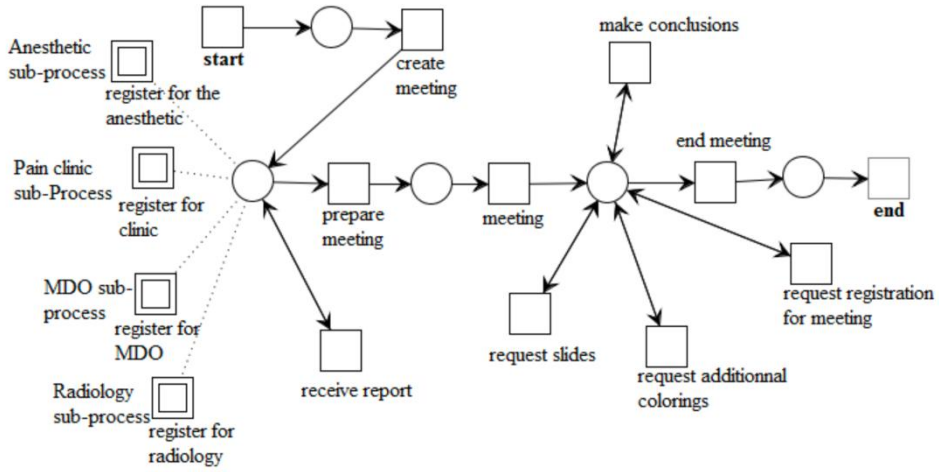

Fig. 7: The pathology proclet

Personal factors have also been established as important factors in process understandability, and are focussed on the interaction between human beings and the computer. It has been difficult to define formal metrics to evaluate this dimension of understandability because it strongly depends on the person (its experience, knowledge, personality, perception capabilities and others).

In addition to the factors mentioned above, there are others factors that can also influence the understandability of a process. Among others, we have: model purpose (the com- prehension of a process may be influenced by the purpose of the modeler); problem domain (people may find easier to understand processes about the domain with which they are familiar); modelling notation (given the plethora of modelling notations (UML activity diagram, BPMN, EPCs, YAWL, Petri nets), some have been proved to be more suitable (BPMN and UML activity diagram) for end-users than others. This work used rather petri nets in the discovery process because of re- search needs. However, the ProM toolkit offers the possibility to convert a petri net notation into a BPMN notation) ; and visual appearance (Mapikou & Etoundi 2016) (the appearance may affect the comprehension of a process because two processes may be visually different but semantically the same). The decomposition used in the framework allows simplifying the visual appearance of the process.

D. Soundness

The proposed framework discovers sub-processes by applying the algorithm alpha R a certain number of times (equal to the number of participants involved in the overall process), so it satisfies at least the quality level offered by the alpha-algorithm; it has been proved in Leemans et al. (2015, 2016) that the alpha-algorithm guarantees soundness on small datasets. Based on the decomposition principle used to split the log in the work, the soundness dimension of the resulting process can be better compared to the existing one. Indeed, after splitting the log into sub-logs, it is easy to ensure that each sub-log is quite smaller than the initial one and can be handled by the alpha-algorithm with guarantee of soundness and ease-to understand. So, based on the size of each sub-log, it is possible to guarantee soundness on each sub-process. It remains to demonstrate the soundness of the overall proclet system.

The theoretical idea developed to establish the soundness of the collaborative system is to assimilate this system to a com- posite system that can be easily modeled using petri nets. In composite systems, if it is possible to demonstrate a property on each component of the system, then by composition Abadi & Lamport (1993), the property can be verified on the overall composite system. More precisely, assimilating the proclet system as a composite system, since we have guaranteed that each sub-process is sound, then by composition, the proclet system is sound.

7. Conclusion

The purpose of this paper was to propose a process discov- ery solution able to deal with the lack of understandability of process models generated by existing process discovery algorithms. It has been identified that the most highly scalable algorithms existing in the literature try to guarantee most of the discovery quality dimensions but resulting process models are not understandable nor interpretable by end-users of the or- ganization. These techniques seemed to be of high importance for scientific research because it provided good performances of computation, but less useful for practitioners who wish to analyse their processes and improve their management.

This paper has proposed a framework that used the divide and conquer approach to discover a particular class of process models called proclets. Proclets have been identified as a process model that can improve understandability of complex processes. The proposed framework has been divided in three steps: decomposition, discovery and interconnection. The decomposition step divides the log into sublogs; the discovery step associates each sublog to a subprocess. For this step, a refinement of the alpha-algorithm named alpha R has been proposed; the interconnection establishes the communication between subprocesses in order to build a unique coherent sys- tem. Experiments have been carried a real log to validate our proposal. As a result, the understandability of the discovered process has been significantly improved. Soundness of the resulting process model has been discussed and theoretically demonstrated.

For future works, it would be interesting to deepen the experiments on the framework in order to allow automating the assessment of soundness and understandability. Furthermore, the problem of lack of understandability on the discovered process model may remain in presence of a large number of resources (participants). Exploring a mechanism to reduce the number of participants in this situation would be interesting.

8. References

[1]. Abadi, M. & Lamport, L. (1993), 'Composing specifications', *ACM Transactions on Programming Languages and Systems (TOPLAS)* 15(1), 73–132.

[2]. Alves de Medeiros, A., Van Dongen, B., Van Der Aalst, W. & Weijters, A. (2004), Process mining: Extending the α-algorithm to mine short loops, Technical report, BETA Working Paper Series.

[3]. Atkinson, C. & Kuhne, T. (2003), 'Model-driven development: a metamodeling foundation', *IEEE software* 20(5), 36–41.

[4]. Bergenthum, R., Desel, J., Mauser, S. et al. (2009), 'Synthesis of petri nets from term based representations of infinite partial languages', *Fundamenta Informaticae* 95(1), 187– 217.
[5]. Bhattacharya, K., Gerede, C., Hull, R., Liu, R. & Su, J. (2007), Towards formal analysis of artifact-centric business process models, *in* 'International Conference on Business Process Management', Springer, pp. 288–304.
[6]. Buijs, J. C., van Dongen, B. F. & van der Aalst, W. M. (2012), A genetic algorithm for discovering process trees, *in* 'Evolutionary Computation (CEC), 2012 IEEE Congress on', IEEE, pp. 1–8.
[7]. Carmona, J., Cortadella, J. & Kishinevsky, M. (2010), 'New region-based algorithms for deriving bounded petri nets', *IEEE Transactions on Computers* 59(3), 371–384.
[8]. Chinosi, M. & Trombetta, A. (2012), 'Bpmn: An introduc- tion to the standard', *Computer Standards & Interfaces* 34(1), 124–134.
[9]. Claes, J. & Poels, G. (2012), Process mining and the prom framework: an exploratory survey, *in* 'International Conference on Business Process Management', Springer, pp. 187– 198.
[10]. de Medeiros, A. K. A., Weijters, A. J. & van der Aalst, W. M. (2007), 'Genetic process mining: an experimental evalua- tion', *Data Mining and Knowledge Discovery* 14(2), 245– 304.
[11]. De Weerdt, J., De Backer, M., Vanthienen, J. & Baesens, B. (2012), 'A multi-dimensional quality assessment of state- of-the-art process discovery algorithms using real-life event logs', *Information Systems* 37(7), 654–676.
[12]. Evermann, J. (2016), 'Scalable process discovery using map-reduce', *IEEE Transactions on Services Computing* 9(3), 469–481.
[13]. Fahland, D., De Leoni, M., Van Dongen, B. F. & Van Der Aalst, W. M. (2011), Conformance checking of inter- acting processes with overlapping instances., *in* 'BPM', Vol. 6896, Springer, pp. 345–361.
[14]. Ferreira, H. M. & Ferreira, D. R. (2006), 'An integrated life cycle for workflow management based on learning and planning', *International Journal of Cooperative Information Systems* 15(04), 485–505.
[15]. Frost, R. A. (1993), 'Guarded attribute grammars', *Software: Practice and Experience* 23(10), 1139–1156.
[16]. Goedertier, S., Martens, D., Vanthienen, J. & Baesens, B. (2009), 'Robust process discovery with artificial neg- ative events', *Journal of Machine Learning Research* 10(Jun), 1305–1340.
[17]. Greco, G., Guzzo, A., Pontieri, L. & Sacca, D. (2006), 'Discovering expressive process models by clustering log traces', *IEEE Transactions on Knowledge and Data Engi- neering* 18(8), 1010–1027.
[18]. Gunther, C. W. & Rozinat, A. (2012), 'Disco: Discover your processes.', *BPM (Demos)* 940, 40–44.
[19]. Gunther, C. W. & Van Der Aalst, W. M. (2007), Fuzzy mining– adaptive process simplification based on multi-perspective metrics, *in* 'International conference on business process management', Springer, pp. 328–343.
[20]. Janssenswillen, G., Donders, N., Jouck, T. & Depaire, B. (2017), 'A comparative study of existing quality measures for process discovery', *Information Systems* 71, 1–15.
[21]. Lamma, E., Mello, P., Montali, M., Riguzzi, F. & Storari, S. (2007), Inducing declarative logic-based models from labeled traces, *in* 'International Conference on Business Process Management', Springer, pp. 344–359.
[22]. Leemans, S. J., Fahland, D. & van der Aalst, W. M. (2013*a*), Discovering block-structured process models from event logs-a constructive approach, *in* 'International Conference on Applications and Theory of Petri Nets and Concurrency', Springer, pp. 311–329.
[23]. Leemans, S. J., Fahland, D. & van der Aalst, W. M. (2013*b*), Discovering block-structured process models from event logs containing infrequent behaviour, *in* 'International Conference on Business Process Management', Springer, pp. 66–78.
[24]. Leemans, S. J., Fahland, D. & van der Aalst, W. M. (2014*a*), Discovering block-structured process models from incom- plete event logs, *in* 'International Conference on Applica- tions and Theory of Petri Nets and Concurrency', Springer, pp. 91–110.
[25]. Leemans, S. J., Fahland, D. & van der Aalst, W. M. (2014*b*), Exploring processes and deviations, *in* 'Inter- national Conference on Business Process Management', Springer, pp. 304–316.
[26]. Leemans, S. J., Fahland, D. & van der Aalst, W. M. (2015),
[27]. Scalable process discovery with guarantees, *in* 'International Conference on Enterprise, Business-Process and Informa- tion Systems Modeling', Springer, pp. 85–101.
[28]. Leemans, S. J., Fahland, D. & van der Aalst, W. M. (2016), 'Scalable process discovery and conformance checking', *Software & Systems Modeling* pp. 1–33.
[29]. Mans, R., Russell, N. C., van der Aalst, W. M., Bakker, P. J., Moleman, A. J. & Jaspers, M. W. (2010), 'Proclets in healthcare', *Journal of Biomedical Informatics, Elsevier* 43(4), 632–649.

[30]. Mapikou, G. L. M. & Etoundi, R. A. (2016), A process mining oriented approach to improve process models analysis in developing countries, *in* 'Computer Systems and Applications (AICCSA), 2016 IEEE/ACS 13th International Conference of', IEEE, pp. 1–8.
[31]. Mendling, J., Neumann, G. & Van Der Aalst, W. (2007), Understanding the occurrence of errors in process models based on metrics, *in* 'OTM Confederated International Con- ferences" On the Move to Meaningful Internet Systems"', Springer, pp. 113–130.
[32]. Mendling, J., Reijers, H. A. & Cardoso, J. (2007), What makes process models understandable?, *in* 'International Conference on Business Process Management', Springer, pp. 48–63.
[33]. Murata, T. (1989), 'Petri nets: Properties, analysis and appli- cations', *Proceedings of the IEEE* 77(4), 541–580.
[34]. Redlich, D., Gilani, W., Molka, T., Drobek, M., Rashid, A. & Blair, G. (2014), Introducing a framework for scal- able dynamic process discovery, *in* 'Enterprise Engineering WorkingConference', Springer, pp. 151–166.
[35]. Redlich, D., Molka, T., Gilani, W., Blair, G. & Rashid, A. (2014*a*), Constructs competition miner: process control- flow discovery of bp-domain constructs, *in* 'International Conference on Business Process Management', Springer, pp. 134–150.
[36]. Redlich, D., Molka, T., Gilani, W., Blair, G. S. & Rashid, A. (2014*b*), Scalable dynamic business process discovery with the constructs competition miner., *in* 'SIMPDA', Citeseer, pp. 91–107.
[37]. Reijers, H. A. & Mendling, J. (2011), 'A study into the factors that influence the understandability of business pro- cess models', *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* 41(3), 449–462.
[38]. Scheer, A.-W. (2012), *Business process engineering: reference models for industrial enterprises*, Springer Science & Busi- ness Media.
[39]. Tiwari, A., Turner, C. J. & Majeed, B. (2008), 'A review of business process mining: state-of-the-art and future trends', *Business Process Management Journal* 14(1), 5–22.
[40]. van Beijsterveld, J. A. & Van Groenendaal, W. J. (2016), 'Solving misfits in erp implementations by smes', *Informa- tion Systems Journal* 26(4), 369–393.
[41]. Van Der Aalst, W. (2011), *Process mining: discovery, con- formance and enhancement of business processes*, Vol. 2, Springer.
[42]. Van Der Aalst, W., Adriansyah, A., De Medeiros, A. K. A., Arcieri, F., Baier, T., Blickle, T., Bose, J. C., Van Den Brand, P., Brandtjen, R., Buijs, J. et al. (2011), Process mining manifesto, *in* 'International Conference on Business Process Management', Springer, pp. 169–194.
[43]. Van der Aalst, W. M. (1998), 'The application of petri nets to workflow management', *Journal of circuits, systems, and computers* 8(01), 21–66.
[44]. Van der Aalst, W. M. (2013), Process mining in the large: a tutorial, *in* 'European Business Intelligence Summer School', Springer, pp. 33–76.
[45]. Van Der Aalst, W. M., Barthelmess, P., Ellis, C. A. & Wainer, J. (2001), 'Proclets: A framework for lightweight interacting workflow processes', *International Journal of Cooperative Information Systems* 10(04), 443–481.
[46]. van der Aalst, W. M., Mans, R. & Russell, N. C. (2009), 'Workflow support using proclets: Divide, interact, and conquer.', *IEEE Data Eng. Bull.* 32(3), 16–22.
[47]. van der Aalst , W. M., Rubin, V., van Dongen , B. F., Kindler, E. & Gu¨nther, C. W. (2006), 'Process mining: A two-step approach using transition systems and regions', *BPM Center Report BPM-06-30, BPMcenter. org* 6.
[48]. Van der Aalst, W. M., Rubin, V., Verbeek, H., van Dongen, B. F., Kindler, E. & Gu¨nther, C. W. (2010), 'Process mining: a two-step approach to balance between underfitting and overfitting', *Software & Systems Modeling* 9(1), 87.
[49]. Van der Aalst, W., Weijters, T. & Maruster, L. (2004), 'Work- flow mining: Discovering process models from event logs', *IEEE Transactions on Knowledge and Data Engineering* 16(9), 1128–1142.
[50]. Van der Werf, J., van Dongen, B. F., Hurkens, C. A. & Serebrenik, A. (2008), Process discovery using integer linear programming, *in* 'Petri Nets', Vol. 5062, Springer, pp. 368– 387.
[51]. Verbeek, H., Buijs, J., Van Dongen, B. & van der Aalst, W. M. (2010), 'Prom 6: The process mining toolkit', *Proc. of BPM Demonstration Track* 615, 34–39.
[52]. Weijters, A. & Ribeiro, J. (2011), Flexible heuristics miner (fhm), *in* 'Computational Intelligence and Data Mining (CIDM), 2011 IEEE Symposium on', IEEE, pp. 310–317.
[53]. Weijters, A., van Der Aalst, W. M. & De Medeiros, A. A. (2006), 'Process mining with the heuristics miner- algorithm', *Technische Universiteit Eindhoven, Tech. Rep. WP* 166, 1–34.

[54]. Wen, L., van der Aalst, W. M., Wang, J. & Sun, J. (2007), 'Mining process models with non-free-choice constructs', *Data Mining and Knowledge Discovery* 15(2), 145–180.
[55]. Wen, L., Wang, J., van der Aalst, W. M., Huang, B. & Sun, J. (2009), 'A novel approach for process mining based on event types', *Journal of Intelligent Information Systems* 32(2), 163–190.