Automated Discovery of Workflow Models from Hospital Data

L.Maruster[1] and **W.van der Aalst**[1] and **T.Weijters**[1] and **A.van den Bosch**[2] and **W.Daelemans**[2]

Abstract. Workflow nets, a subclass of Petri nets, are known as attractive models for analysing complex business processes. In a hospital environment, for example, the processes show a complex and dynamic behavior, which is difficult to control; the workflow net which models such a complex process provides a good insight into it, and due to its formal representation offers techniques for improved control. We propose a method whose main advantage consists in discovering the workflow Petri nets automatically from process logs. We illustrate the functioning of our method on simulated hospital process logs, containing information about medical actions over time. The results of our experiments indicate that this method is able to discover processes whose underlying models are acyclic and sound WF nets, involving parallel, conditional and sequential constructs. We argue that solutions have to be found for cyclic and free-choice/non-free-choice workflow nets.

1 Introduction

Today, the managing of complex business processes calls for the development of powerful information systems, able to control and support the flow of work. These systems are called Workflow Management Systems (WfMS), where a WfMS is generally thought of as "a generic software tool which allows for definition, execution, registration and control of workflows" [1]. Petri nets are attractive as the underlying model language for WfMS: they have a precise mathematical formalism, they provide a graphical image of the investigated processes, they can express all important features of the WfMS, and they can be subject to many analysis techniques [1]. Petri nets used for workflow process definition are called workflow nets (WF nets).

However, the process of workflow design takes a lot of time and the resulting models are often incomplete and unrealistic. In the hospital domain, for example, some projects were developed to support patient WfMS built on guidelines [3],[6],[10]. Specifying the clinical practice in terms of guidelines, which provides the process logic, presupposes the presence of expert knowledge. The knowledge extraction process is time consuming and may not fully reveal the clinical practice. In contrast, hospital processes are highly dynamic and subject to change. Thus, the WfMS has to be flexible enough and able to capture all changes that occurred in a short time frame. Flexible workflows have received a lot of attention, which is also reflected in the research efforts seen in this area [2],[9]. We think that a WfMS able (i) to acquire process knowledge automatically and (ii) to incorporate changes quickly, will be more desirable in the hospital domain and other dynamic workflow environments.

In this paper we present a procedure for discovery a business process, given a workflow log (a "history" which contains information about how events took place, chronologically ordered). The procedure also builds the associated workflow net. The obtained workflow net can be used for analysing, redesigning and managing the investigated process. The idea of discovering models from process logs was previously investigated in contexts such as software engineering processes and also workflow management. Cook and Wolf propose three methods for process discovery in case of software engineer processes: a finite state-machine method, a neural network and a Markov approach [4]. Their methods focuse on sequential processes. Also, they have provided some specific metrics for detection of concurrent processes, like entropy, event type counts, periodicity and causality [5]. Herbst and Karagiannis used a hidden Markov model in the context of workflow management, in the case of sequential processes [8] and concurrent processes [7]. The drawback of Herbst and Karagiannis results is that workflow net constructs like AND/OR splits and joins are not depicted. In the works mentioned, the focus was on identifying the dependency relations between events. Our goal is to detect explicitly (i) flow and (ii) concurrency/choice relations between events. Moreover, and this is our main contribution, we try to discover complete WF nets, not only the dependency relations. WF nets can be used further to analyse the considered process.

Our goal is to develop an automatic discovery and analysis tool for providing insight into real world situations. We take the modeling of logistical processes of medical actions in a hospital. Medical treatment in a hospital often requires involvement of different specialties. Because of an aging population that shows complex syndromes and the increased specialisation of medical technology, the number of different specialties involved in treatments is increasing. We will call a patient that requires different specialties for her/his treatment a medical multidisciplinary patient (MMP). We focus especially on the patient category with vascular medical problems, because the managing of vascular patients involves the largest number of specialties [11]. An efficient coordination of such MMP may imply restructuring the organization of hospitals into specialty-oriented units, while care for patients is not constrained within the boundaries of one of those units [11]. The problem that arises here is how to build these specialty-oriented units. For this purpose, we need to investigate the underlying patient flow process and to decide on the organizational structure of the hospital. In this paper, we concentrate only on developing a tool for investigating the workflow. In future work, we want to apply this tool to our real hospital MMP data.

The structure of the paper is as follows. Section 2 addresses some basic theoretical aspects of WfMS and WF nets. In Section 3 we present our process discovery method. Section 4 summarizes the experiments done for testing our method. A discussion of the status

[1] Eindhoven University of Technology, 5600 MB Eindhoven, The Netherlands
[2] Tilburg University, LE 5000 Tilburg, The Netherlands

and shortcomings of the present approach, and future directions of our work are presented in Section 5.

2 Workflow models and WF nets

Because of their good theoretical foundation, Petri nets (PN) have been used successfully to model and analyse processes from many domains, like for example, software and business processes. Workflow processes can be modeled by WF nets, which form a subclass of PN [1]. A classical Petri net is a directed graph with two kinds of nodes, places and transitions, where arcs connect a place to a transition or a transition to a place. Each place can contain zero, one or more tokens. The state of a classical PN is determined by the distribution of tokens over places. A transition can fire if each of its input contains tokens. If the transition fires, i.e. it executes, it takes one token from each input place and puts one token on each output place.

Workflows are case oriented, which means that each activity executed in the workflow corresponds to a case. In our hospital domain, a case corresponds with a patient and an activity corresponds with a medical activity. The process definition of a workflow assumes that a partial order exists between activities, establishing the execution order of the activities. Referring to the Petri net formalism, workflow activities are modeled as transitions and the causal dependencies between activities are modeled as places and arcs.

A WF net is a classical PN with one source place (i.e. a place without incoming arcs), that represents the beginning of the case in the workflow, and a sink place (i.e. a place without outgoing arcs), which represents the end of the case in the workflow. Each transition and place in the WF net is on a path from source place to sink place. The existence of one token in the source place will correspond to the situation in which a patient is first admitted to the hospital and needs to be registered. One token in the sink place means that the patient registration to that hospital has ended.

The routing in a workflow assumes four kinds of routing constructs: sequential, parallel, conditional and iterative routing [1]. Sequential routing concerns ordered causal relationships between tasks. For example, if we consider tasks A and B, we have a sequential routing construct when task B is executed only after task A is executed. Parallel routing is used when the order of execution is less strict. A parallel routing is modeled by AND-split and AND-join blocks. An AND-split corresponds to a transition with two or more output places and an AND-join corresponds to a transition with two or more input places. Conditional routing allows the modeling of a choice between two or more alternatives. To express the conditional construct, OR-split and OR-join blocks are used. An OR-split corresponds to two or more alternative output transitions and an OR-join corresponds to two or more alternative input transitions. Figure 1illustrates the workflow process definition for handling a medical complaint. In this figure we can identify the following routing constructs: transitions *identification* and *cardiologist* are AND-splits, *I diag OK*, *I diag NOK* and *decide surgery* are AND-joins, c4, c5, c8 and c10 are OR-splits and c6, c9 and c11 are OR-joins.

Our method assumes the discovery of WF nets which are sound. A WF net is sound if and only if: (i) a case can be always completed, (ii) after the completion of an activity, no work is left behind in the workflow and (iii) there are no dead activities, i.e. states that cannot be reached [1]. Of special interest are free-choice Petri nets. A PN is free-choice if and only if for every two transitions that share the same input place, the two corresponding input sets are the same [1].

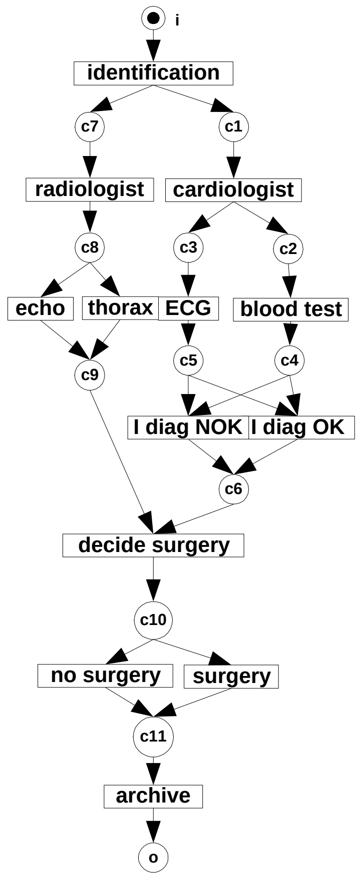

Figure 1. A Petri net for handling a medical complaint.

3 Process discovery method

The present work concentrates on discovering WF models from workflow logs that can be represented as WF nets. The workflow log (WL) contains for each case the related events, in the order in which they took place. A hospital workflow log contains medical activities performed on each patient, as they happen over time. In case of our multidisciplinary patients (MMP) problem, WLs contain sequences of identifiers; each identifier corresponds to a department (e.g. cardiology, radiology, vascular laboratory, etc.) visited by the patient at a specific point in time. Formally, we define a WL as following:

Definition 1 *(Workflow Log WL):*

Let A be a set of observable actions. A *workflow log* is a set of sequences over A, i.e., $WL \subseteq A^*$, where A^* is the set of all sequences that are composed of zero or more actions of A.

Given the WF net process definition from Figure 1, an example of a patient trace is given below:

```
identification, cardiologist, blood_test, ECG,
radiologist, echo, I_diag_OK, decide_surgery,
no_surgery, archive.
```

Consider now the WF net from Figure 1, which is a simplified model of handling a medical complaint. A patient who enters the hospital is first identified with his/her hospital card. Afterwards, he/she has to visit a cardiologist and radiologist, in any order. After visiting the cardiologist, it is necessary to perform both ECG and blood test, in any order. Depending on *ECG* and *blood test* results, a first diagnosis is made. The first diagnosis can relate to a vascular disease problem (*I diag NOK*) or not (*I diag OK*). When the first diagnosis results and either the *thorax* or *echo* results become available, a decision on surgery is made. Depending on this decision, the surgery is performed or not. Finally, this particular patient case is archived.

In this work we address the following research question: given a WL, discover the underlying WF net that generates all events in WL and distinguish its routing constructs. Our discovery technique is based on the following definitions:

Definition 2 *(α, β, first, last):*

For any sequence $s \in A^*$, with $s = (a_1, a_2, ..., a_n)$, we have:

- $\alpha(s) = \{a_1, a_2, ..., a_n\}$ is the alphabet of s,
- $\beta(s) = \{(a_1, a_2), (a_2, a_3), ..., (a_{n-1}, a_n)\}$ is the set of pairs of s,
- $first(s) = a_1$,
- $last(s) = a_n$.

These definitions formalize the concept of trace of events from A^* (the set of all potential traces). Identifiers ai represent events in traces. Traces have a first and a last event. Our algorithm focuses on pairs of events from $WL \subseteq A^*$. Definition 3 states the possible relations that can exist between the elements of a pair. Namely, if there is a sequence in WL, where event y appears after x and there is no sequence where y appears before x, then the pair $(x, y) \in R^{\rightarrow}$. If there is a sequence in WL, where event x appears after y and also y appears after x, then $(x, y) \in R^{\longleftrightarrow}$.

Definition 3 *($R^{\rightarrow}, R^{\longleftrightarrow}$):*

Let WL be a workflow log over A. We consider the following relations:

$$\begin{aligned} R^{\rightarrow} &= \{(x, y) \in A \times A \mid \exists s \in WL, (x, y) \in \beta(s) \wedge \\ &\quad \forall s \in WL, (y, x) \notin \beta(s)\}, \\ R^{\longleftrightarrow} &= \{(x, y) \in A \times A \mid \exists s \in WL, (x, y) \in \beta(s) \wedge \\ &\quad \exists s \in WL, (y, x) \in \beta(s)\}. \end{aligned}$$

The first step for discovery the WF net is to build the net N_0^{WL}, as it is formally stated in the following definition.

Definition 4 *(N_0^{WL})*

Let WL be a workflow log, and $R^{\rightarrow}$ and $R^{\leftrightarrow}$ as defined. Then
$N_0^{WL} = (P, T, F)$, where
$T = \bigcup_{s \in WL} \alpha(s)$,
$P = R^{\rightarrow} \cup \{i, o\}$,

$$\begin{aligned} F = \{(i, t) \in (P \times T) \mid \exists s \in WL, t = first(s)\} \cup \\ \{(t, o) \in (T \times P) \mid \exists s \in WL, t = last(s)\} \cup \\ \{((x, y), t) \in R^{\rightarrow} \times T \mid y = t\} \cup (t, (x, y)) \in T \times R^{\rightarrow} \mid x = t\}. \end{aligned}$$

The net N_0^{WL} can be thought as a "preliminary" net of the real WF net. N_0^{WL} is constructed as following: the place i (the "future" source place in the WF net) links all events from the set $first(s)$ and the place o (the "future" sink place in the WF net) links all events from the set $last(s)$. For all pairs of events $(x, y) \in R^{\rightarrow}$, which have in common the same x, an arc will link the node x with all nodes y. The node x will be the starting point and the nodes y will be the ending points. For all pairs of events $(x, y) \in R^{\rightarrow}$ which have in common the same y, an arc will link all nodes x with node y. The nodes x will be the starting points and the node y will be the ending point. For the rest of pairs of events $(x, y) \in R^{\rightarrow}$, an arc will link node x with node y. The set of all nodes will form T (the set of transitions in the net N_0^{WL}). Additionally, on each arc that connects two nodes, a place from set P (the set of places in the net N_0^{WL}) will be placed. All arcs that connect nodes will form the set F in the net N_0^{WL}.

Definition 5 *(merge):*

Let $N = (P, T, F)$ be a Petri net and $X \subseteq P$ a set of places. Then
$merge(N, X) = (P', T, F')$, where
$P' = (P \setminus X) \cup \{P_X\}$,
$F' = F \cap (P' \times P') \cup \{(P_X, t) \mid t \in T \wedge \exists p \in X, (p, t) \in F\} \cup$
$\{(t, P_X) \mid t \in T \wedge \exists p \in X, (t, p) \in F\}$.

This formalizes the operation of merging two places into one new place Px. The arcs from F will connect a transition from T with the new merged place Px (or the new merged place Px with a transition from T). Intuitively, the ingredients of the merging operation are one "source" transition, two arcs that link the source transition with the two places, and two more arcs that link each place with one of the two "destination" transitions. The result of merging is one "source" transition, one arc that links the "source" transition with the new merged place, and two arcs that link the new merged place with "destination" transitions. The merge operation works analogously in case of two "source" transitions and one "destination" transition.

Definition 6 (N^{WL}):

Let WL be a workflow log, N_0^{WL} and $R^{\rightarrow}$ as defined. Take N_0^{WL} and merge all places that have non-concurrent input or output transitions. The resulting net is N^{WL}. Formally:

$$\begin{aligned} M(N) := \{&(P_1, P_2) \in P \times P \mid (P_1 \bullet \cap P_2 \bullet \neq \varnothing \wedge \\ &\exists t_1 \in \bullet P_1, \exists t_2 \in \bullet P_2, (t_1, t_2) \notin R^{\leftrightarrow}) \vee \\ &(P_1, P_2) \in P \times P \mid (\bullet P_1 \cap \bullet P_2 \neq \varnothing \wedge \\ &\exists t_1 \in P_1 \bullet, \exists t_2 \in P_2 \bullet, (t_1, t_2) \notin R^{\leftrightarrow})\}, \end{aligned}$$

for any Petri net $N = (P, T, F)$.

In the above definition, $\bullet P_i$ means the set of input transitions for place P_i, and $P_i \bullet$ means the set of transitions sharing P_i as an input place. The decision to merge places that have non-concurrent input or output transitions is taken if the input transitions are in relation $(t_1, t_2) \notin R^{\leftrightarrow}$, or if the output transitions are in relation $(t_1, t_2) \notin R^{\leftrightarrow}$, too.

For building the WF net N^{WL}, we have to apply the following algorithm:

$N := (P, T, F) = N_0^{WL}$;
while $M(N) \neq \varnothing$
do $(P_1, P_2) :\in M(N)$
$N := merge(N, \{P_1, P_2\})$
od
$N^{WL} := N$.

This algorithm states that first, the net N_0^{WL} is built (see Definition 4). Second, for each pair $(P_1, P_2) \in M(N)$, the decision to merge the related places is made.

To illustrate our algorithm, we use the WF net presented in Figure 1. Suppose that we have a WL corresponding to the WF from Figure 1, and we want to discover the underlying WF net. The basic idea of our approach supposes three main steps:

- Identify the pairs $(x, y) \in R^{\rightarrow}$ and $(x, y) \in R^{\leftrightarrow}$ (Definition 3);

In WL, the pairs $(x, y) \in R^{\rightarrow}$ are those pairs of events that always occur in the same order, for example: *(identification, cardiologist), (identification, radiologist), (cardiologist, ECG), (cardiologist, blood_test), (I diag OK, decide surgery), (I diag NOK, decide surgery), (radiologist, thorax), (radiologist, echo), (thorax, decide surgery), (echo, decide surgery)*, and so on. The pairs $(x, y) \in R^{\leftrightarrow}$ are pairs of events that can happen in any order, like, for example, *(cardiologist, radiologist), (ECG, blood_test), (cardiologist, thorax)*, and so on.

- Using pairs $(x, y) \in R^{\rightarrow}$, build the net N_0^{WL} (Definition 4);

In this step the pairs N_0^{WL} are connected and a place is inserted between the connected transitions. The result of this step is shown in Figure 2.

- Apply the algorithm for building the WF net N^{WL}, which merges all places that have non-concurrent input or output transitions (Definition 5 and 6).

In Figure 2, because (*I diag OK, I diag NOK*)$\notin R^{\leftrightarrow}$, we have to merge places 4 with 4'. The same states for 5 with 5', 6 with 6', 8 with 8', 9 with 9', 10 with 10' and 11 with 11'. After merging, we get the WF net from Figure 1.

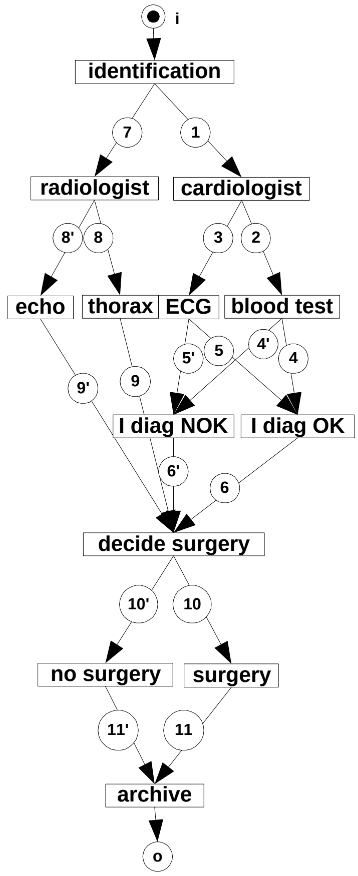

Figure 2. The N_0^{WL} net for handling a medical complaint.

4 Experiments

We tested our method, considering five different sound and acyclic WF nets. The WF nets are similar with the example given in Figure 1, i.e. they contain between 10-12 transitions, involving parallel, conditional and sequence routing constructs. For each WF net, we generated random workflow logs with 500 event traces.

In four experiments, the WF net built with our method is equivalent to the initial WF net. In the left side of Figure 3 the initial WF net of one of our experiments is presented. In the right side of Figure 3 is shown the N_0^{WL} net. In the N_0^{WL} net, after merging places 2 and 2', 7 with 7' and 10 with 10', we obtained the initial WF net. In the other three experiments, the WF nets considered have comparable structure and complexity.

However, in the fifth experiment involving a WF net which is actually not free-choice, the method was not able to find the complete WF net.

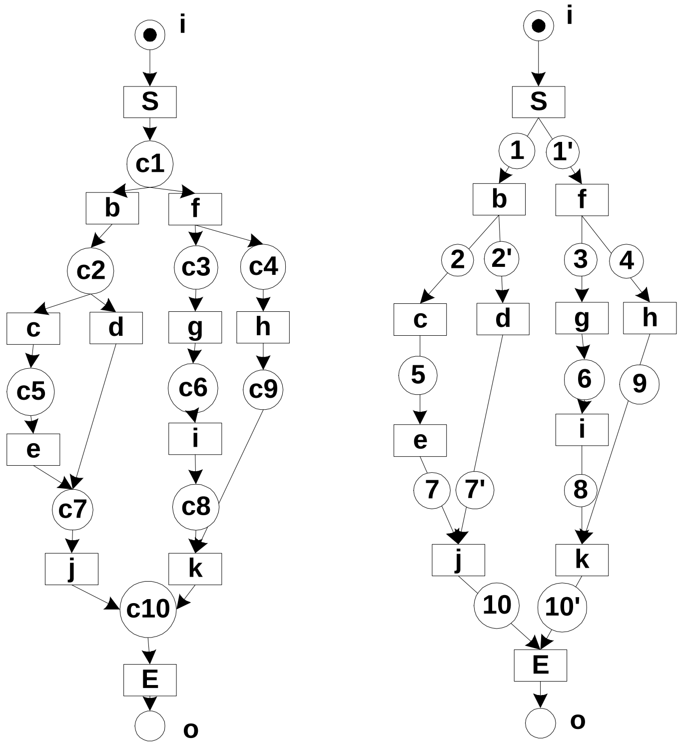

Figure 3. One example of sound and acyclic WF net. After merging places in the figure from the right side, the resulting WF net is identical with the initial WF net from the left side.

5 Discusson and future work

In this work we have presented a method for discovering the underlying WF net from a process workflow log. The experiments done with five different WF nets show that in the case of sound and acyclic WF nets, involving parallel, conditional and sequential constructs, our method is able to rebuild them correctly from their workflow logs. However, the current technique does not work for all kinds of WF nets, as one experiment involving a non-free-choice net showed. We need to carry out further experiments to determine the types of WF nets where our method is applicable and to provide theoretical foundations.

Our research is preliminary; we plan to do future research in several directions. First, we want to extend our method to cyclic WF nets (we investigated the performance of our method in the acyclic case). Especially in the medical domain, follow-up visits to the same specialist happen very often; thus the detection of the iteration in a process is necessary. Second, we want to improve our method so that it can also be applied to free-choice and also non-free-choice WF nets.

Our final goal is to have a robust tool which is able to discover and further analyse a complex and completely unknown process, namely the logistical flow of medical multidisciplinary patients (MMP). Such a tool coupled with WfMS that offer generic modeling and enactment capabilities can provide an efficient way of analysing and managing today's business organisations.

REFERENCES

[1] W.M.P Aalst, 'The application of petri nets to workflow management', *Journal of Circuits, Systems, and Computers*, **8(1)**, 21–66, (1998).

[2] W.M.P. Aalst and eds. Jablonski, S., 'Flexible workflow technology driving the networked economy', *International Journal of Computer Systems, Science, and Engineering*, **15(5)**, (2001).

[3] S. Andersen. Patman: Patient workflow management system. EU Project, 1998.

[4] J.E. Cook and A.L. Wolf, 'Discovering models of software process from event-based data', *ACM Transactions on Software Engineering and Methodology*, **7(3)**, 215–249, (1998).

[5] J.E. Cook and A.L. Wolf, 'Event-based detection of concurrency', in *Proc. of the Sixth International Symposium on the Foundations of Software Engineering*, pp. 35–45, (1998).

[6] L. Dazzi, C. Fassino, R. Sarraco, S. Quaglini, and M. Stefanelli, 'A patient workflow management system built on guidelines', in *Proc. of AMIA 97, Nashville*, pp. 146–150, (1997).

[7] J. Herbst, 'Dealing with concurrency in workflow induction', in *Proc. of European Concurrent Engineering Conf., SCS Europe, Gent, Belgium*, (2000).

[8] J. Herbst, 'A machine learning approach to workflow management', *Lecture Notes in Computer Science*, **1810**, 183–194, (2000).

[9] M. Klein, C. Dellarocas, and eds. Bernstein, A., 'Adaptive workflow systems', *Computer Supported Cooperative Work*, **9(3-4)**, (2000).

[10] S. Quaglini, M. Stefanelli, G. Lanzola, V. Caporusso, and S. Panzarasa, 'Flexible guideline-based patient careflow systems', *Artificial Intelligence in Medicine*, **22(1)**, 65–80, (2001).

[11] G.G. Vries. Multidiciplinaire samenwerking rondom vaatpatienten. Tussenrapportage, Prismant, 2001.