A Novel Approach for Process Mining: Intentional Process Models Discovery

Ghazaleh Khodabandelou, Charlotte Hug, Camille Salinesi

Toward an Automatic Tool for the Construction of Intentional Process Models from Event Logs

Ghazaleh Khodabandelou
Centre de Recherche en Informatique
University Paris 1 Panthéon-Sorbonne
90 Rue Tolbiac, Paris 75013, France
ghazaleh.khodabandelou@malix.univ-paris1.fr

Charlotte Hug
Centre de Recherche en Informatique
University Paris 1 Panthéon-Sorbonne
90 Rue Tolbiac, Paris 75013, France
charlotte.hug@univ-paris1.fr

Camille Salinesi
Centre de Recherche en Informatique
University Paris 1 Panthéon-Sorbonne
90 Rue Tolbiac, Paris 75013, France
camille.salinesi@univ-paris1.fr

***Abstract*—So far, process mining techniques have suggested to model processes in terms of tasks that occur during the enactment of a process. However, research on method engineering and guidance has illustrated that many issues, such as lack of flexibility or adaptation, are solved more effectively when intentions are explicitly specified. This paper presents a novel approach of process mining, called Map Miner Method (MMM). This method is designed to automate the construction of intentional process models from process logs. MMM uses Hidden Markov Models to model the relationship between users' activities logs and the strategies to fulfill their intentions. The method also includes two specific algorithms developed to infer users' intentions and construct intentional process model (Map) respectively. MMM can construct Map process models with different levels of abstraction (fine-grained and coarse-grained process models) with respect to the Map metamodel formalism (*i.e.*, metamodel that specifies intentions and strategies of process actors). This paper presents all steps toward the construction of Map process models topology. The entire method is applied on a large-scale case study (Eclipse UDC) to mine the associated intentional process. The likelihood of the obtained process model shows a satisfying efficiency for the proposed method.**

I. Introduction

Processes are tightly coupled to information systems supporting them. The strong relation between processes and supporting infrastructures permit a fast-spreading availability of event logs [1]. The logs provide a basis for what is actually happening in processes. Process mining field has emerged a few years ago as a key approach to design business processes from event logs [2], [3], [4]. Process mining techniques use the data logs to discover actual process, to measure the compliance of the real processes with respect to the prescribed ones, and thereby improve processes design, methods and tools. Process Mining endeavors to gain insight into various perspectives, such as the process perspective, the performance and organizational perspectives. Whereas most process mining approaches specify behaviors in terms of sequences of tasks and branching, research on intention-oriented process modeling has shown that the fundamental nature of processes is mostly intentional. Intentions as a first class concept of information systems engineering [5] have appeared in the early 80s, in the information systems community [6], [7] as a *potential theoretical foundation* to determine user's behavior [8]. Intention models take root in a former work, called Technology Acceptance Model (TAM) [8], one of the extensions of Theory of Reasoned Action (TRA) [9] designed to model humans' behavioral intention, particularly computer usage behavior. The TRA has proven effective in predicting and explaining human behavior through various domains. Since the early 90s, intention-oriented process models have been promoted as a driving paradigm to study strategic alignment [10], to define actors and roles, to model organizational changes [11], to specify the outcome of business process models [12] and name them, to analyze, to support guidance [13], [14], to describe intentional services [15], to handle traceability issues [16], to express pervasive information systems [17], to study users behavior to identify and name use cases, to tailor methods [18] or just make them more flexible [19], etc. Further, research on guidance in method engineering shows that many method engineering issues, such as rigidity or lack of adaptation, are solved more effectively when intentions and strategies are explicitly specified [5].

Many works on intention-oriented modeling indicate how to express intentions, formalize them in models, relate them with other concepts, analyze them to solve a series of information systems engineering issues, such as several scenario-based techniques [20]. However, intention-oriented process modeling has largely neglected event logs so far. These event logs recorded by information systems contain precious information about the actual enacted processes. They can be used to provide an insight into the actual processes; deviations can be analyzed and the quality of models can be improved. Nevertheless, the major known difficulty is: how to identify and formalize the intentions from event logs?

Discovery of intentional process models from event logs has been proposed for the first time in our previous works [21], [22], [23]. The lack of an automatic method to identify and formalize users' intentions from event logs of the process enactment motivates the contribution of this paper. In this perspective, this paper presents a method, so-called Map Miner Method (MMM) that aims at discovering intentions and strategies from event logs and thereby building the actual Map process model. MMM can be useful at different stages of the process model life-cycle, for instance: (i) at the requirements level, to find users intentions from processes activity logs; and (ii) at the project management level, to check alignment

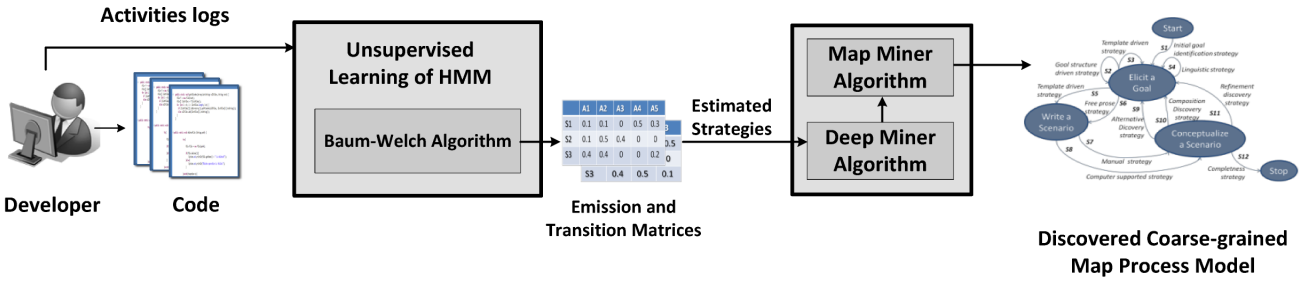

Fig. 1. An overview of Map Miner Method

between prescribed process models and what stakeholders actually do. MMM generates intentional models specified with the Map formalism [5]. This formalism is chosen rather than other intentional process models such as KAOS or I* [24], [25] because (a) it allows process variability and multi-process specification [26], (b) it has already proved effective for specifying software engineering processes [13], and (c) it combines intentions and strategies at multiple levels of abstraction, which scales well to large and complex processes [5]. Merely, Map process model [26] has proved a powerful tool to better understand the deep nature of processes, to see how processes interweave and combine, to abstract processes and visualize them under man-manageable form, even when they are extremely complex [5].

In the MMM framework, the nature of intentions highly depends on the definition of intention in the Map formalism. In this formalism, an intention is defined as a goal, an objective or a motivation to achieve with clear-cut criteria of satisfaction, which can be fulfilled by the enactment of a process [27]. Furthermore, the intentions form the high-level goals (*e.g.*, organizational goals) and are explicitly represented in the models.

The contribution of this paper consists of: (i) first, modeling users' strategies in terms of observed activities (event logs) using Hidden Markov Models (HMMs) [28], (ii) Second, using estimated strategies and Map formalism, we developed two particular algorithms (Deep Miner and Map Miner algorithms) to generate respectively, fine-grained and coarse-grained Map process models, (iii) Finally, the proposed approach (MMM) is applied on a large-scale case study, **Eclipse UDC** (Usage Data Collector) developers' logs [29]. The resulting Maps provide valuable information about the processes followed by the developers and demonstrate the effectiveness and scalability of MMM.

The remainder of this paper is organized as follows. Section II introduces MMM, by presenting first how to model strategies in terms of activities with HMM, then the Deep Miner and Map Miner algorithms are represented. Section III represents a large-scale case study with event logs from Eclipse UDC to find out how developers use the *Eclipse technology*. Related works are discussed in section IV and threats to validity in section V. Finally, section VI concludes this work and presents its perspectives.

II. Map Miner Method

The key idea of MMM is modeling the users' behaviors in terms of their underlying intentions and strategies (*i.e.*, the alternative ways to fulfill their intentions) from event logs. MMM uses the Map formalism to model actual processes followed by users. According to the Map formalism, **intentions** express what users intend to perform during the enactment of a process [26]. In other words, the enactment of a process is execution of a sequence of **activities** that are caused by users' intentions. According to the fuzzy mechanism of cognitive processes, an intention causes the performance of one or several activities at time t. However, intentions can be fulfilled by combining different activities, which are different ways of achieving an intention (*i.e.*, strategies). Indeed, in the Map metamodel, strategies are used to move from one intention to another. These strategies in turn, are made of one or several activities. This relationship between the intentions, the strategies and the activities, represents the top-down reasoning and acting structure of cognitive processes of human brain. Nevertheless, only the low-level part of this structure, *i.e.*, users' activities, is observable. The high-level part (strategies and intentions) is an abstract notion and therefore unobservable directly.

MMM proposes to trace back this structure to discover the source, *i.e.*, users' intentions. To do so, MMM uses the observable users' activities traces (a set of event logs), generated while interacting with information systems. MMM consists of three phases: (i) First phase: estimating the users' strategies from observed activities using HMMs, (ii) Second phase: generating fine-grained Map process model using estimated strategies and Map formalism (Deep Miner algorithm), (iii) Final phase: generating coarse-grained Map process model from fine-grained one (Map Miner algorithm).

MMM generates the high-level intentions to obtain the coarse-grained Map and also the low-level intentions to obtain the fine-grained Map. These low-level intentions are so-called *sub-intentions*. They are the finest intentional objects of the Map metamodel. Each sub-intention is associated to a parent

intention, and one intention is fulfilled if at least one of its children sub-intention is fulfilled.

This multi-level topology is due to the deep architecture of the brain. The extensive studies on the visual cortex show each sequence of cortex zones contains a representation of the input and also signals flow from one to the other [30]. In other words, each level of this feature hierarchy represents the input at a different level of abstraction, with more abstract features further up in the hierarchy, defined in terms of the lower-level ones. Therefore, cognitive processes have a deep structure and humans organize their ideas and concepts hierarchically. First, they learn simpler concepts and then compose them to represent more abstract ones [30].

Figure 1 depicts an overview of MMM. Constructing the Map process model thereby allows rebuilding the actual process model, *i.e.*, the model followed by users. Before addressing the problem of discovering the Map, we briefly explain its framework.

A. The Map Metamodel

Map is an intentional process metamodel. Map process model (an instance of Map metamodel) allows representing process models in terms of users' intentions and strategies. Figure 2 illustrates a Map process model where the nodes represent the intentions and the edges represent the strategies. A set of *<Source Intention, Strategy, Target Intention>* represents a *section* in the Map. Map allows representing flexible process models, enacted in a dynamic way since the sections of a Map can be executed non-sequentially and as long as intentions are not completely fulfilled. For example, on figure 2, one way to fulfill the intention *Specify an entity* is to select the strategy S_4: *By generalization*. Thus, confronted to a specific situation and a particular intention, the Map reveals the alternative strategies to fulfill the intention. The corresponding section of this example is *<Specify an entity, By generalization, Specify an entity>*. In the following section, we explain how HMMs allow modelling strategies with respect to the activities from event logs.

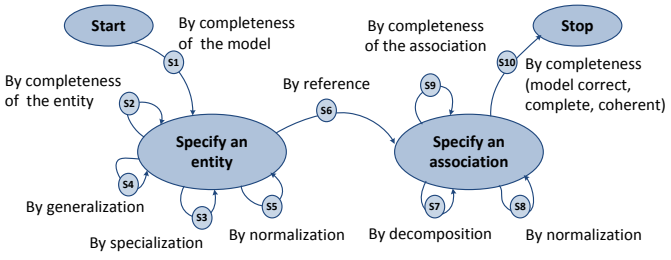

Fig. 2. An example of Map process model with 4 intentions and 10 strategies

B. Estimating Hidden Strategies from the Observed Activities

HMMs allow modeling the structure of complex temporal dependencies between two complementary Markov processes: hidden and observed processes, given that the observed process is generated depending on the state of the hidden process. Generally, the states of hidden process are not visible but the probability of a given state can be inferred by computing the Maximum Likelihood of the observed process. It turns out that the topology of HMMs is particularly adapted to model the relation between strategies and activities in the Map formalism. To make it clear, let us consider an example for a Map process model enacted with 2 strategies and an HMM realized with 2 hidden states (see figure 3). As shown this figure, strategies are used to move from one intention to another and are made of one or several activities. For instance, the strategy 1 allows moving from intention *a* to intention *b* and it is made of activities a_1, a_3 and a_4. The same structure can be found in an HMM, where hidden states are processed sequentially and generate observations. For instance, hidden state 1 generates the activities a_1, a_3 and a_4. This similar topology motivates using HMMs to model users' strategies with respect to activity logs. The topology (the order of the

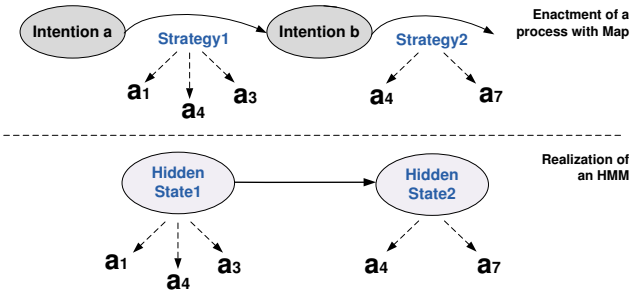

Fig. 3. An example of a Map process model enacted with 2 strategies (above) and an HMM realized with 2 hidden states (below)

Markov process) for MMM framework is formally defined next.

1) Hidden Process as Users' Strategies: The hidden process represents the users' strategies. Let $\mathbf{s} = (s_1, \ldots, s_L) \in \mathcal{S}^L$ be a temporal sequence of users' strategies of length L. The topology M_1 is chosen for this process, which means that the strategy s_l at step l only depends on the strategy at step $\ell - 1$. This choice is justified by the fact that strategies are performed in a logical order by users. A homogeneous Markov chain, which parameters are denoted by $\mathbf{T}$ and π, models transitions between strategies with:

$$\begin{aligned} \mathbf{T}(u, v) &= \Pr\left(s_\ell = v | s_{\ell-1} = u\right) \ \forall u, v \in \mathcal{S}, \ \ell \in [2, L], \\ \pi(u) &= \Pr\left(s_1 = u\right) \ \forall u \in \mathcal{S}. \end{aligned} \tag{1}$$

The vector π contains the probabilities of strategy at the initial state and the matrix $\mathbf{T}$ contains the transition probabilities for the following strategies, *i.e.*, the transition probabilities from any strategy at step $\ell - 1$ to any other strategy at step ℓ (including itself).

2) Observed Process as Users' Activities: The observed process represents the users' activities. Let $\mathbf{a} = (a_1, \ldots, a_L) \in \mathcal{A}^L$ be a temporal sequence of users' activities of length L. We choose the model M_0 for this process, meaning that the emission of a_ℓ, at a given step ℓ, does not depend on any

previous activity. It only depends on the strategy at the same time step. The emission probability of an activity $a \in \mathcal{A}$ for a given strategy $u \in \mathcal{S}$ is given by:

$$\mathbf{E}(a, u) = \Pr(a|u). \tag{2}$$

The matrix $\mathbf{E}$ contains the emission probabilities of any activity for any strategy.

3) Learning HMMs Parameters: Assuming that $\mathcal{S}$, $\mathcal{A}$ and π are known, the HMM model parameters is fully described by $\mathcal{H} = \{\mathbf{E}, \mathbf{T}\}$, which represents the core information about the HMM behavior. Since these matrices generate the sequences of strategies and the observed sequences of activities, they can be estimated from the sequences of activities. This approach is known as unsupervised learning and present the desirable feature of characterizing strategies based only on activities. The Baum-Welch algorithm (BWA) [31] is most commonly used to learn the parameters of a HMM.

The BWA estimates the HMM parameters that locally maximize the probability of having the sequences of activities generated by the HMM. More precisely, the BWA maximizes the likelihood of $\mathcal{H}$:

$$\mathcal{H}^* = \arg\max_{\mathcal{H}} \prod_{n=1}^{N} \Pr(\mathbf{a}_n|\mathcal{H}), \tag{3}$$

where N is number of observed sequences in a dataset containing activities $\mathbf{a}_1, \ldots, \mathbf{a}_N$.

C. Determining the Number of Strategies

The BWA requires the sets $\mathcal{A}$ and $\mathcal{S}$ to be known or at least, their cardinality, *i.e.* $|\mathcal{S}|$ and $|\mathcal{A}|$ for the algorithm to run. Regarding the set of activities $\mathcal{A}$, it can simply be obtained by identifying the different activities in the dataset. However, obtaining the set of strategies $\mathcal{S}$ is impossible since there is no information about strategies in the dataset. There are three ways to obtain the number of strategies:

- This parameter can be chosen by experts. This is interesting since it allows to set the level of complexity of the model, to meet some a priori expectations of the model. However, as this choice involves human intervention and introduces a bias.
- Several criteria exist to determine the number of hidden states, such as Bayesian Information Criterion (BIC) [32]. This metric allows the comparison of HMM models with different numbers of hidden states, trained on the same underlying data. BIC penalizes the likelihood of the model by a complexity factor proportional to the number of parameters θ in the model and the number of training observations $\mathcal{R}$:

  $$BIC = -2\log(\Pr(\mathcal{A}|\mathcal{H})) + \theta\log(\mathcal{R}), \tag{4}$$

  where $\theta = (J^2 + J \times F)$, and J^2 and $J \times F$ represent the number of parameters in transition matrix and emission matrix, respectively. Although BIC can ensure a result for every sequence of activities, this trade-off does not allow generating the model with the best likelihood when the model has a high complexity factor.
- The method that is used in this paper to set the right number of strategies is heuristic. It consists in generating several HMM models with different numbers of strategies and observing the associated emission matrices. It occurs that as the number of possible strategies increases, the number of different strategies obtained in the emission matrices reaches a threshold. It means that when the number of possible strategies is too high, the BWA produces an emission matrix with several identical strategies. Consequently, we choose to set the right number of strategies of our model to this observed threshold. This method has the advantage of being adaptable for different datasets. The drawback of this method is its computation complexity.

D. Deep Miner Algorithm

Once model parameters $\mathbf{T}$ and $\mathbf{E}$ are estimated by the BWA, the problem is how to extract a Map process model, which fits actual process model. As a first step into this direction, we propose a metric which has the interesting property of taking into account both fitness and precision to optimize the Map process model, whereas classical metrics in process modeling address either fitness or precision (see [2] for an overview of the existing metrics).

1) Extracting a Map from a Transition Matrix: We recall that the matrices generated by the BWA are an emission matrix $\mathbf{E}$, giving the probabilities of generating any activity while performing a strategy and a transition matrix $\mathbf{T}$, giving the probabilities of transition between any couple of strategies $(s, s') \in \mathcal{S}^2$.

Clearly, there is a strong link between the transition matrix and the topology of the Map process model we want to extract. To extract a Map from a transition matrix, the two following constraints must be verified: (i) any transition between possible strategies in the transition matrix should be possible on the Map, (ii) any transition between possible strategies in the Map should be possible in the transition matrix.

The first constraint can be seen as a criterion for fitness since it ensures that all the transitions learned from the dataset are present in the Map. The second constraint corresponds to a criterion of precision since it aims at avoiding introducing extra-transitions in the Map that are not learned from the dataset. Our goal is to find the Map that best satisfies both of them. In the next part, we define a metric which is a trade-off between fitness and precision and also captures the relative importance of transitions. Figure 4 depicts an overview of Deep Miner algorithm.

2) Proposed Metric of Fitness and Precision: The topology of a Map $\mathbf{m}$ can be defined by the set of its sections, each comprising a source sub-intention, a strategy and a target sub-intention. We formally write

$$\mathbf{m} = (m_k)_{k \in \{1,\ldots,K\}}, \tag{5}$$

where k denotes the index of a section and K is the total number of sections of the Map. For each $k \in \{1, \ldots, K\}$, $m_k = (i, s, j) \in \mathcal{I} \times \mathcal{S} \times \mathcal{I}$. The component $m_k(1)$ is the

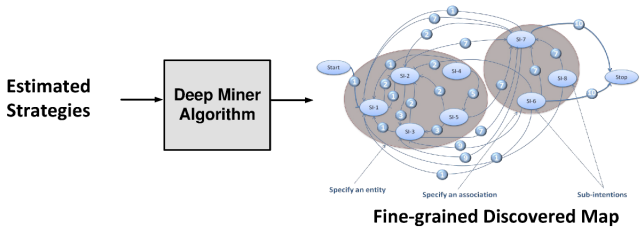

Fig. 4. Overview of Deep Miner Algorithm.

source sub-intention of section k, $m_k(2)$ is the strategy of section k, and $m_k(3)$ is the target sub-intention. On the Map $\mathbf{m}$, a transition from strategy s to strategy s' is possible if and only if there exist $(k, k') \in \{1, \ldots, K\}^2$ such that $m_k(2) = s$, $m_{k'}(2) = s'$, and $m_k(3) = m_{k'}(1)$. In the following, we use the symbol α to denote if a transition is possible or not in the Map:

$$\alpha_{s,s'} = \begin{cases} 1 \text{ if } \exists (k,k') \in \{1,\ldots,K\}^2 \text{ such that } m_k(2) = s, \\ m_{k'}(2) = s', \text{ and } m_k(3) = m_{k'}(1), \\ 0 \text{ otherwise.} \end{cases} \tag{6}$$

In the transition matrix $\mathbf{T}$, we only consider as valid transitions with a probability above a given threshold ε. The value of ε has to be chosen heuristically, to counter the effects of noise and artifacts in the dataset. We define:

$$\omega_{s,s'} = \begin{cases} 1 \text{ if } \mathbf{T}(s,s') \geqslant \varepsilon, \\ 0 \text{ if } \mathbf{T}(s,s') < \varepsilon. \end{cases} \tag{7}$$

Classically, the criteria of fitness and precision between $\mathbf{T}$ and $\mathbf{m}$ can be expressed by the expressions known as recall and precision. In our context, we define these two expressions as

$$\text{Rec}(\mathbf{T}, \mathbf{m}) = \frac{\sum_{s,s'} \omega_{s,s'} \alpha_{s,s'}}{\sum_{s,s'} \omega_{s,s'}}, \tag{8}$$

$$\text{Pre}(\mathbf{T}, \mathbf{m}) = \frac{\sum_{s,s'} \omega_{s,s'} \alpha_{s,s'}}{\sum_{s,s'} \alpha_{s,s'}}. \tag{9}$$

The numerator of both expressions is the number of significant transitions in $\mathbf{T}$ that are present on the map $\mathbf{m}$, while the denominators are the number of significant transitions in $\mathbf{T}$ and the number of transitions on $\mathbf{m}$, respectively.

Since our goal is to find a map that fits best the transition matrix with respect to both recall and precision, we can use the classical F-measure which expression is:

$$F_1(\mathbf{T}, \mathbf{m}) = 2 \frac{\text{Pre}(\mathbf{T}, \mathbf{m}) \text{Rec}(\mathbf{T}, \mathbf{m})}{\text{Pre}(\mathbf{T}, \mathbf{m}) + \text{Rec}(\mathbf{T}, \mathbf{m})}. \tag{10}$$

3) Optimization Problem: Now that the proper metric has been defined, we need to find the Map that maximizes it. The solution of this problem belongs to the set

$$\mathcal{M} = \arg\max_{\mathbf{m}} F_1(\mathbf{T}, \mathbf{m}). \tag{11}$$

Since we want to obtain a Map with the simplest structure, we choose the solution with the lowest number of sections. In other words, the solution is

$$\mathbf{m}^* = \arg_{\mathbf{m} \in \mathcal{M}} \min |\mathbf{m}|, \tag{12}$$

where $|\mathbf{m}|$ stands for the number of sections in $\mathbf{m}$. However, finding $\mathbf{m}^*$ is a difficult task since $\mathbf{m}$ generally belongs to a high-dimension space. Indeed, it can be shown that there are $2^{|\mathcal{S}|^2}$ possible Maps for $|\mathcal{S}|$ different strategies. Consequently, computing all the possible Maps with a brute force method then comparing their F-measures is not an option. Instead, we developed an algorithm that solves (12) with a complexity bounded by $|\mathcal{S}| * (|\mathcal{S}| - 1)$. This algorithm is detailed below.

Data: strategy set $\mathcal{S}$, transition matrix $\mathbf{T}$, threshold ε
Result: map $\mathbf{m}^*$
for *each strategy* $s \in \mathcal{S}$ **do**
  associate to s a target sub-intention i_s;
end
for *each strategy* $s \in \mathcal{S}$ **do**
  for *each strategy* $s' \in \mathcal{S},\ s' \neq s$ **do**
    if $\mathbf{T}(s, s') \geqslant \varepsilon$ **then**
      create a section from i_s to $i_{s'}$ with strategy s' ;
    end
  end
end

Algorithm 1: How to obtain a Map from $\mathcal{S}$, $\mathbf{T}$, and ε.

The first part of algorithm 1 associates a target sub-intention to each strategy of $\mathcal{S}$. In the second part, if a transition probability from strategy s to strategy s' is above the threshold ε, a section is added to the Map from the target sub-intention of s to the target sub-intention of s'. This section ensures that the transition given by $\mathbf{T}$ is also present in the Map. With this algorithm, recall and precision, defined in (8) and (9), have the advantage of being equal to 1. Indeed, ε defines the abstraction-level in the Map. When ε is close to 0, almost all the transitions from the unsupervised model are present in the obtained Map. Consequently, the likelihood of the obtained Map is high but the Map is hardly understandable by humans since it has too many sections. However when ε increases, the number of sections, as well as the likelihood of the obtained Map, decrease. The Map gets more easily understandable by humans but it is not as accurate in terms of transition.

E. Map Miner Algorithm

Granularity refers to the level of detail of a process model. While a Map process model with a *coarse-grained* granularity represents high-level intentions (*e.g.*, organizational intentions), a Map process model with a *fine-grained* granularity provides more detailed intentions, called sub-intentions. Depending on the situation at hand, one can define the nature of granularity that is needed. This affects the kind of guidance, explanation and trace that can be provided [33].

For instance, project manager or middle managers require rather coarse-grained process description as they want to gain an overview of time, budget, and resource planning for their

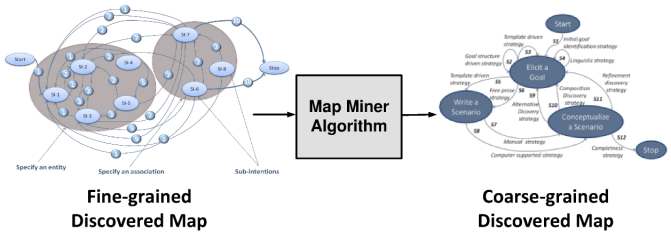

Fig. 5. Overview of Map Miner Algorithm.

decisions. In contrast, software engineers, users, testers, analysts, or software system architects will prefer a fine-grained process model where the details of the model can provide them with instructions and important execution dependencies such as the dependencies between people. Hybrid formalisms such as Process Weaver [34] uses different notations for coarse-grained and fine-grained aspects of process.

1) Determining the Level of Abstraction for the Map: Given that the Map obtained from MMM has a high degree of precision (in terms of sub-intentions and sections), it can be useful to extract some higher-level Maps of the same process from this original Map. An algorithm has been developed to automatically perform this task and is presented in this section. It falls into three main parts:

1) The sub-intentions from the fine-grained Map are represented in a space in which they can be classified into clusters.
2) A clustering algorithm, namely *K-means*, is applied on the sub-intentions in order to group them into clusters of intentions. Note that the number of intentions is a parameter that has to be chosen. The choice of this parameter allows researchers to obtain Maps with different level of precision.
3) Finally, a Map is rebuilt from the new groups of intentions with updated sections.

2) Sub-intentions Representation in the Space: Before clustering sub-intentions into groups of intentions, one needs to represent each sub-intention in a space that trustfully accounts for the topology around the sub-intention in the original Map. Given that each sub-intention is connected to other sub-intentions by sections, a proper way to represent sub-intentions is to indicate to which other sub-intentions it is connected. Since the Map is an oriented graph, a difference is made between sub-intentions from which there exists a strategy going to the sub-intention to be represented and sub-intentions that can be fulfilled with strategies from the sub-intention to be represented.

From a formal perspective, let us consider a Map with N sub-intentions. The sub-intention $i_n \in \mathcal{I}$ is represented by a vector v_n in a space of dimension $2N$ such that the first N coefficients correspond to the sub-intentions from which there is a strategy going to i_n, and the final N coefficients correspond to the sub-intentions that can be fulfilled from i_n.

- For all $n' \in [1, N]$, if there exists a section from $i_{n'}$ to i_n then $v_n(n') = 1$, otherwise $v_n(n') = 0$.
- For all $n' \in [1, N]$, if there exists a section from i_n to $i_{n'}$ then $v_n(2n') = 1$, otherwise $v_n(2n') = 0$.
- Since i_n is implicitly considered to be connected to itself, we have $v_n(n) = 1$ and $v_n(2n) = 1$.

Let us consider a fragment of a fine-grained Map with 8 sub-intentions (figure 6). The sub-intentions 'a', 'b', $\cdots$,'h' are respectively represented by the vectors v_1, v_2,$\cdots$, v_8 in the space. For instance, we consider the vectors v_3 and v_5, which represent 'c' and 'e', respectively.

$$v_3 = (01110000 \quad 00100110)$$
$$v_5 = (01011000 \quad 00001110)$$

These vectors are composed of the values '0' or '1'. The vectors sizes are equal to 16; this size is due to possibilities of 8 incoming sub-intentions 'a', 'b', $\cdots$,'h' and 8 outgoing sub-intentions 'a', 'b', $\cdots$,'h' (comprising the sub-intention it-self for both cases). The left side of the vectors defines the incoming sub-intentions and the right side defines the outgoing sub-intentions. For instance, in v_5 the incoming sub-intentions are 'b', 'd' and 'e' and the outgoing sub-intentions are 'e', 'f' and 'g'. Therefore, in v_5 the value of '1' is assigned to the incoming sub-intentions 'b', 'd', 'e' (at the left side) and the outgoing sub-intentions 'e', 'f', 'g' (at the right side). The value '0' is assigned to the other sub-intentions. In the same way, in v_3 the incoming sub-intentions are 'b', 'c' and 'd' and the outgoing sub-intentions are to 'c', 'f' and 'g'. Therefore, in v_3 the value of '1' is assigned to the incoming sub-intentions 'b', 'c', 'd' (at the left side) and the outgoing sub-intentions 'c', 'f', 'g' (at the right side) and the value '0' to the other sub-intentions.

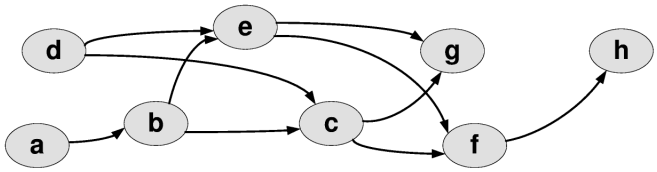

Fig. 6. A fragment of a fine-grained Map with 8 sub-intentions

With this representation, two sub-intentions connected to similar sub-intentions will be represented with a short distance between them, while two sub-intentions connected to different sub-intentions will be represented with an important distance between them. This way, the clustering algorithm that is applied on the sub-intentions can group sub-intentions efficiently. Figure 5 represents an overview of Map Miner Algorithm.

3) Clustering sub-intentions into high-level intentions: Once the sub-intentions are represented in the space described in section II-E2, a clustering algorithm can be applied to group them into clusters of intentions. In this work, we apply the *K-means* [35] algorithm to perform this task. This algorithm works the following way: given a number K of clusters and a set of points $(v_n)_{n\in[1,N]}$, the algorithm determines the gravity center c_k of each cluster k, such that the sum of the distances from the points to the center of their cluster is minimized. In

other words, it minimizes the sum

$$\sum_{1:N} d\left(v_n, c(v_n)\right), \tag{13}$$

where $c(v_n)$ is the center of gravity which is the closest to v_n. For example, if v_n is closer to c_k, $c(v_n) = c_k$. And $d(.,.)$ is a distance between two points.

Said in another way, the *K-means* algorithm finds K groups of sub-intentions such that in each group, sub-intentions are in a same area of the Map.

4) Rebuilding the Map: To obtain a new Map from the clusters of intentions, all the previous sub-intentions are replaced by the intention of their group. The sections also need to be updated to take into account the simplified topology of the Map. We recall that the Map discovered by the Deep Miner algorithm is denoted by $\mathbf{m}^*$. Note that the identical sections have to be removed from the discovered Map. Algorithm 2 shows how to rebuild a coarse-grained Map process models from K clusters of intentions given that the mapping from sub-intentions to intentions obtained from K-means is denoted by g.

Data: Map $\mathbf{m}^*$, mapping g
Result: Map $\widetilde{\mathbf{m}}$
for *each section* m_u^*, $u \in [1, U]$ **do**
  $\widetilde{m}_u(1) := g\left(m_u^*(1)\right)$
  $\widetilde{m}_u(2) := g\left(m_u^*(2)\right)$
end
Remove identical sections in $\widetilde{\mathbf{m}}$

Algorithm 2: Rebuilding a Map from K clusters of intentions.

III. Case study: Eclipse UDC

This case study aims at reconstructing the Map process model of Eclipse Usage Data Collector (UDC) developers [29]. They committed their code to a server hosted by *Eclipse Foundation*. Eclipse UDC strives for helping developers and organizations to better understand how the community uses Eclipse platform [36]. In this perspective, our contribution is to model the UDC developers' behaviors in terms of intentions and strategies while using Eclipse platform. The obtained Map can help to better understand developers' behaviors.

The dataset contains $1,048,576$ event logs from developers who agreed to send their data back to the Eclipse Foundation. These data aggregate activities from over $10,000$ Java developers between September 2009 and January 2010. The activities are recorded by timestamps for each developer, which allows knowing when and by whom activities were committed.

A. Applying MMM on Dataset

In order to apply MMM, it is important to prepare the dataset. The number of unique developers' activities per month exceeds 500 activities. This number contains both the recurring activities and the non-recurring activities, *i.e.*, activities which are not frequently performed by developers. These activities are not representative of the developers' behavior characteristics because they have not been repeated enough to be a behavioral-pattern. For this reason, and also for readability, we limit this study to the 150 most frequent activities performed by developers.

Table I contains the list of these activities. Some of these activities are the commands performed directly by developers; some of them are the frameworks, plug-ins or built-in features of Eclipse used by developers during their development process. For readability reasons, the prefix *org.eclipse* of the activities is removed. The plug-ins and frameworks are shown in bold letters and the related activities are inside brackets.

Once the dataset is ready, BWA estimates the developers' strategies. Note that, the number of strategies obtained by the heuristic method for this case study is 10. The strategies are represented in table I with their corresponding groups of activities. Finally, the Maps obtained by Deep Miner and Map Miner algorithms are shown on figure 8 and 9, respectively.

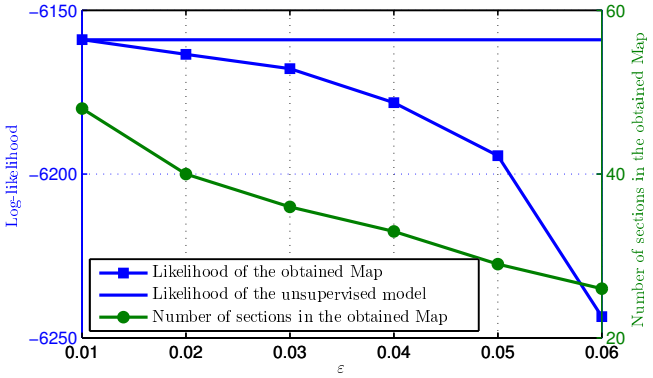

Fig. 7. The effect of the choice of ε on the likelihood and the number of sections of the obtained Map for the Eclipse traces.

Figure 7 depicts the effect of the choice of ε on the likelihood and the number of sections of the obtained Map. As mentioned earlier, ε expresses the level of abstraction for a Map. An expert can choose the value of ε regarding the expected level of abstraction. In this case study, the value of ε is set to 0.06 to have a good trade-off between having a likelihood with a relative high value and a reasonable number of sections.

Regarding the obtained Map (figures 8 and 9), 22 sub-intentions are grouped by Map Miner algorithm into 7 groups of high-level intentions. Note that MMM can discover accurately the beginning and the end of a process; thus the intentions **Start** and **Stop** are clearly determined on the obtained Map. The transition probabilities from one intention to a strategy are annotated on the arrows. These values correspond to the probabilities that the developers selected a strategy from a given intention. The values on the loops indicate the probabilities that the developers continued to perform the activities related to the looped strategies.

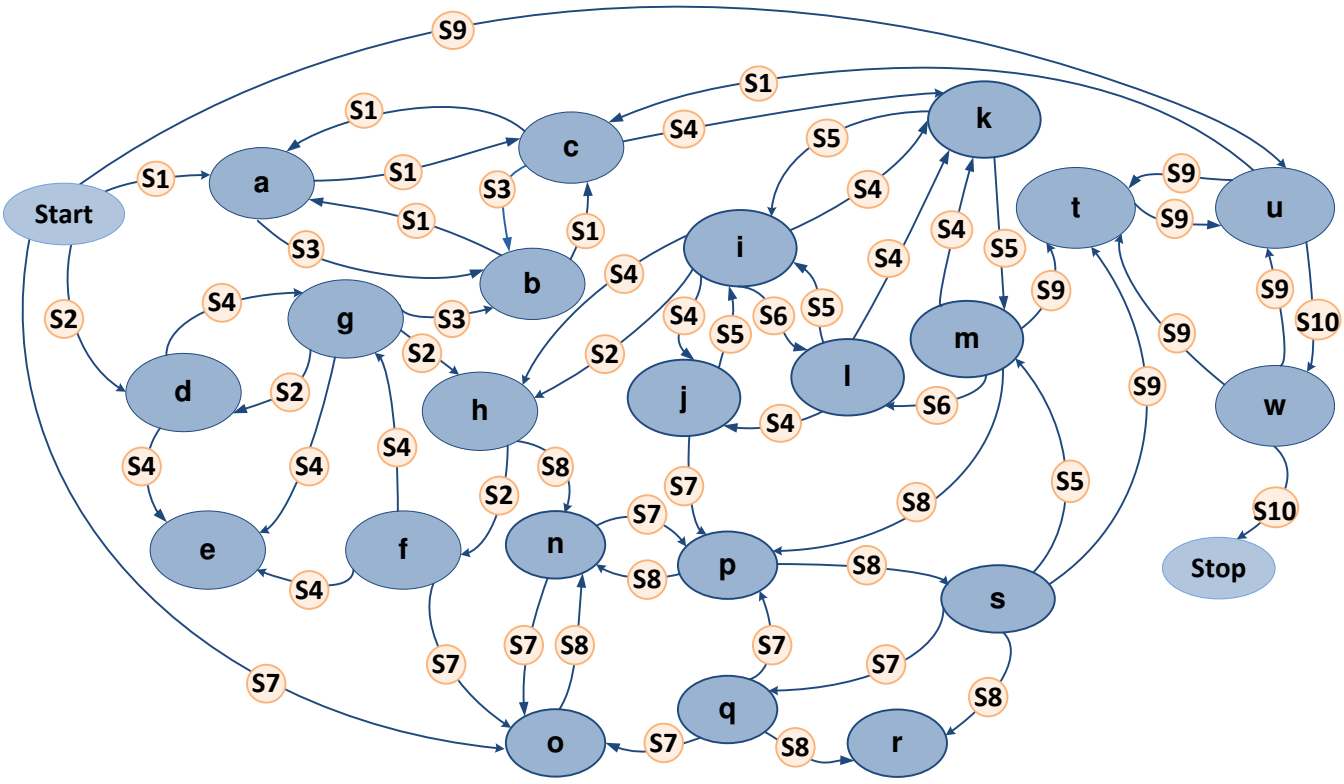

Fig. 8. Fine-grained Map process model obtained with Deep Miner algorithm

TABLE I
STRATEGIES LABELS AND RELATED ACTIVITIES FOR ECLIPSE UDC

Strategies labels	Activities names
S_1	**mylyn.context.ui.commands.**[Open.context.dialog, AttachContext, interest.Increment, interest.Decrement], **mylyn.monitor.ui**, **mylyn.team.ui**, **mylyn.tasks.ui.commands.**[OpenTask, AddTaskRepository, ActivateTask, SearchForTask]
S_2	**core.**[jobs, net, filesystem, resource, runtime, variables, contenttype, databinding.observable], **equinox.p2.ui.sdk.install**
S_3	**mylyn.context.ui.commands.**[Open.context.dialog, AttachContext, interest.Increment, interest.Decrement],**team.cvs.ui.**[branch, replace, GenerateDiff, ShowHistory, Add, Tag,merg, compareWithTag], **jsch.core**, **mylyn.**[monitor.ui, team.ui, commons.ui, bugzilla.ui]
S_4	**pde.ui.**EquinoxLaunchShortcut.run, **equinox.p2.ui.**sdk.update, **equinox.**[ds, simpleconfigurator.manipulator, frameworkadmin, app, common, directorywatcher, engine, core, metadata.repository, garbagecollector, ui.sdk.scheduler, repository, preferences, exemplarysetup, registry, updatechecker]
S_5	**core.**[databinding.observable, core.net, core.filesystem, core.resource, core.runtime, core.variables, core.contenttype], **debug.ui.commands.**[RunLast, Debuglast, eof, StepOver, TerminateAndRelaunch, execute, AddBreakPoint, TogglebreakPoint], **jdt.debug.ui.** [commands.Execute, commands.Inspect], **jdt.junit.**[junitShortcut.rerunLast, gotoTest, junitShortcut.debug], **ltk.ui.refactoring.commands.**[deleteResources, renameresources, moveResources], compare.selectPreviousChange
S_6	**ui.edit.**[delete, paste, copy, undo, text.goto.lineEnd, text.contentAssist.proposals, text.goto.wordNext], **ui.file.save**
S_7	cdt.ui.editor, **jdt.junit.**[junitShortcut.rerunLast, gotoTest, junitShortcut.debug], **team.cvs.ui.**[CompareWithRevision, CompareWithLatestRevisionCommand, CompareWithWorkingCopyCommand],**ui.edit.**[delete, paste, copy undo, text.goto.lineEnd, text.contentAssist.proposals, text.goto.wordNext]
S_8	**team.ui.**[synchronizeLast, teamSynchronizingPerspective, synchronizeAll, applyPatch], **ltk.core.refactoring.refactor.**[create.refactoring.script, show.refactoring.history]
S_9	**mylyn.monitor.ui**, **mylyn.context.ui**, **mylyn.commons.ui**, **team.cvs.ui.**[commitAll, Commit, CompareWithRemote, Sync]
S_{10}	**mylyn.monitor.ui**, **mylyn.bugzilla.core**, **mylyn.bugzilla.ui**, **team.cvs.ui.**[commitAll, Commit, CompareWithRemote, Sync]

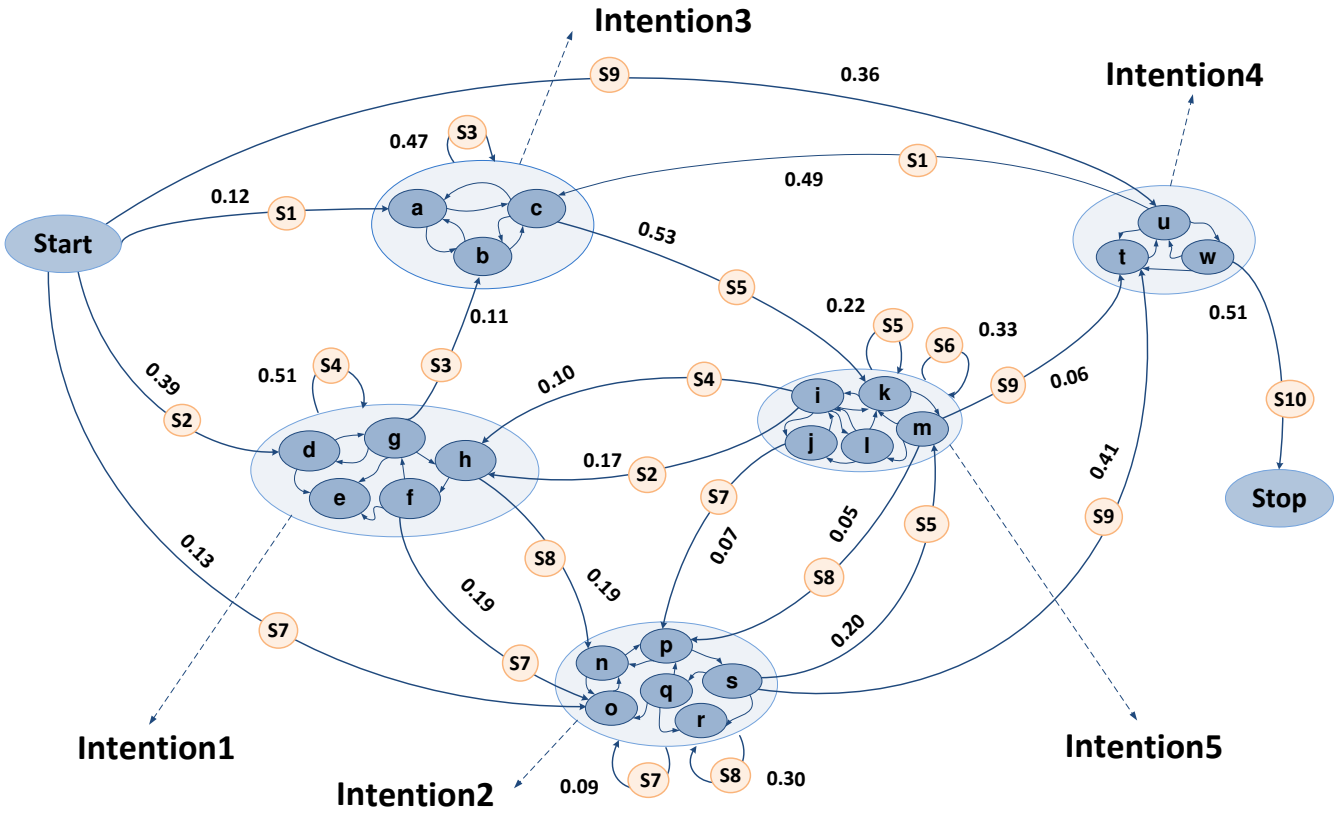

Fig. 9. Coarse-grained Map process model obtained with Map Miner algorithm

TABLE II
STRATEGIES LABELS, TOPICS AND INFERRED STRATEGIES NAMES FOR UDC ECLIPSE

Strategies labels	**Topics obtained by E**	**Inferred strategies names**
S_1	OpenTask, AddTaskRepository, ActivateTask, SearchForTask, Open.context.dialog, AttachContext, interest.Increment, interest.Decrement, mylyn.monitor.ui, mylyn.team.ui	By project tracking and team planning
S_2	jobs, net, filesystem, resource, runtime, variables, contenttype, databinding.observable, equinox.p2.ui.sdk.install	By regular programming activities
S_3	Open.context.dialog, AttachContext, interest.Increment, interest.Decrement, branch, replace, GenerateDiff, ShowHistory, Add, Tag,merg, compareWithTag, jsch.core, monitor.ui, team.ui, commons.ui, bugzilla.ui	By code/task sharing
S_4	EquinoxLaunchShortcut.run, equinox.p2.ui.sdk.update, simpleconfigurator.manipulator, frameworkadmin, app, common, directorywatcher, engine, core, metadata.repository, garbagecollector, ui.sdk.scheduler, repository, preferences, exemplarysetup, registry, updatechecker	By OSGI-based design
S_5	databinding.observable, core.net, core.filesystem, core.resource, core.runtime, core.variables, core.contenttype, RunLast, Debuglast, eof, StepOver, TerminateAndRelaunch, execute, AddBreakPoint, TogglebreakPoint, debug.commands.Execute, commands.Inspect, junitShortcut.rerunLast, gotoTest, ltk.ui.refactoring.commands. deleteResources, renameresources, moveResources, compare.selectPreviousChange	By refactoring, testing and debugging
S_6	delete, paste, copy, undo, text.goto.lineEnd, text.contentAssist.proposals, text.goto.wordNext, ui.file.save	By file modification
S_7	cdt.ui.editor, junitShortcut.rerunLast, gotoTest, delete, paste, copy, undo, text.goto.lineEnd, text.contentAssist.proposals, junitShortcut.debug, CompareWithWorkingCopyCommand, team.cvs.ui.CompareWithRevision, CompareWithLatestRevisionCommand	By reviewing and testing
S_8	synchronizeLast, TeamSynchronizingPerspective, synchronizeAll, applyPatch, create.refactoring.script, show.refactoring.history	By patch applying
S_9	mylyn.monitor.ui, mylyn.context.ui, mylyn.commons.ui, team.cvs.ui.commitAll, Commit, CompareWithRemote, Sync	By CVS committing
S_{10}	mylyn.monitor.ui, mylyn.bugzilla.core, mylyn.bugzilla.ui, team.cvs.ui.commitAll, Commit, CompareWithRemote, Sync	By updating issue tracking

B. Analysis of Obtained Maps

Discovering the Map for developers of Eclipse UDC allows understanding the developers' behaviors during the development process. As shown by figure 9, they have selected different paths (sequences of strategies) with different probabilities to fulfill their intentions. An expert can analyze these behaviors in order to understand how, why and with which probabilities developers make use of different components or plug-ins of Eclipse: where they follow the *best practice* of software development projects and where they deviate from these rules, which components or plug-ins are more involved than the others, which paths are more/less taken or where are system bottlenecks, etc. The Map can also be used to provide recommendations to developers in order to choose the best path to fulfill his/her intentions.

C. Strategies and Intentions Naming Procedure

As found in this case study, MMM constructs the topology of the Map process model from event logs, which means strategies (the arrows) and where they lead, *i.e.*, intentions (the nodes). However, the names of strategies and intentions remain to be inferred. As a first step to this direction, this paper propose to use the information found in the emission matrix $\mathbf{E}$. Indeed, $\mathbf{E}$ contains the names of activities related to each estimated strategy. This can be used as a basis for the construction of ontologies rules.

Table II illustrates the topics found by $\mathbf{E}$ for each estimated strategy as well as manually inferred names for the strategies. The inferred names of strategies are manually inferred through a semantic analysis of the topics found by $\mathbf{E}$, their properties and interrelationships. For instance, the main activities grouped into the strategy S_{10} are 'mylyn.bugzilla.core', 'mylyn.bugzilla.ui', and 'CompareWithRemote'. From these activities one can infer the developers who performed these activities wanted to update the issues on the Bugzilla bug-tracker. Therefore, the name inferred for this strategy is *By updating issue tracking*. Further, the main activity for strategy S_9 is 'team.cvs.ui.commitAll', which means the developers wanted to commit their code to a CVS (Concurrent Versions System) server; thus, the name inferred for this strategy is *By CVS committing*. In the same way, since the strategies lead to intentions, the names of intentions can be inferred by analyzing the strategies leading to each intention. In this paper, we do not infer the names of intentions, we assign to them only the labels, such as **Intention 1**, **Intention 2**, $\cdots$. The names of intentions as well as the strategies remains to be fully automated by building sophisticated ontologies.

IV. Related work

Process mining approaches propose to discover process models from event logs generated during process enactment [3], [4], [37], [38]. The idea to apply process mining in workflow processes context was introduced by Agrawal [39]. At the same time, Datta proposed to discovery of business process models [40]. Cook et al. investigated similar issues in the context of software engineering processes [37]. The majority of the process mining techniques focus on process models discovery based on observed event logs [3], [39], [37], [40], [41], [42], [43]. Process mining techniques are not limited to only process models discovery, for instance, social networks and other organizational models can be discovered from event logs [44], [45]. An overview of the early work in this domain is given in [46].

Inference methods infer process models with a tradeoff between accuracy and noise robustness. Cook compares in [47] three inference algorithms of RNet [48], Ktail [49] and Markov models [50] for process discovery. The latter two are considered as the most promising approaches. Markov models is a hybrid approach (*i.e.*, a statistical and algorithmic approach), looking at the neighboring past behavior to define the future state. It is robust to noise with a controllable complexity. Later, Cook and Wolf in [51], [52] proposed some techniques for concurrency detection and a measure to quantify the variance between behaviors and process models. α-algorithm [3] was proposed by Van der Aalst et al. to rebuild the causality in the Petri nets workflow from the existent relations in the event log. α-algorithm takes the event logs as input, rebuilds process models by using simple XOR, AND splits and joins; thereby creates the workflow nets as output. α-algorithm cannot handle noise and certain complicated routing constructs of workflow nets, such as loops and long-term dependencies, particularly during complex situations [1]. Another approach is directed acyclic graphs [39], which proposes to transform the events into dependency graphs or workflow graphs using directed acyclic graph, representing events and their causal relations without loop. The approach of Agrawal deals with problems of finding a workflow graph creating event logs and defining the edge conditions. However, using this kind of graphs to model the processes is delicate as loops exist in process models. To tackle this challenge, this approach tries to count the tasks frequencies and then fold the graph. Nevertheless, the results are partially satisfying and the model does not completely fit the actual process. Approach of genetic algorithm [53] provides process models (Petri nets) built on causal matrix, *i.e.*, input and output dependencies for each activity. This technique tackles problems such as noise, incomplete data, non-free-choice constructs, hidden activities, concurrency, and duplicate activities. Nevertheless, it requires the configuration of many parameters to deal with noise and irrelevant data, which is a complex task. Inductive workflow acquisition [54] aims at finding the best Hidden Markov Models (HMMs) [28] that reflect the process models acquisition out of workflow models as well as their adaptation to requirements changes. This consists of two steps: induction and transformation steps. This approach supports appearing the same tasks at several times in the model (duplicate tasks). It is similar to the approach of directed acyclic graphs due to the presence of the splits and joins in the transformation step. However, all process mining techniques aim at modeling users' behaviors in terms of activities and overlook the underlying humans' cognitive operators, such as intentions and strategies. A similar technique to HMMs is Finite State Machine (FSM), which

is used in some works to model users' behaviors [49], [38]. While our approach models hidden states as users' strategies, FSM is a simple automaton and is not appropriate to model cognitive processes.

V. Threats to Validity

Four main issues may threat the validity of the proposed approach. First, the quality of discovered Map process models depends on the number of activities sequences used to estimate the parameters of the HMM. Indeed, these sequences have to capture all the possible behaviors while enacting the process under study to produce an accurate Map. If this number is not high enough, the discovered Map process models may suffer from underfitting problems. Second, for the complex data the BWA requires an important number of iterations to converge to a local optimum. For instance, it converges at $20,237$ learning iterations for the Eclipse case study. Moreover, it cannot always be guaranteed to converge to the global maximum likelihood. Third, currently the most sophisticated topology of HMMs that allowing the use of an algorithm, such as BWA or equivalent algorithms, is the M_1M_0 topology. However, this topology is not always the more appropriate to model some processes. Nevertheless, there is no known algorithm to estimate the parameters of the more complex topologies. For this reason, the scope of MMM is limited to the M_1M_0 topology. Fourth, as mentioned earlier, although the MMM automatically discovers the topology of the Map process models, the names of strategies and intentions are still inferred semi-automatically. Inferring these names manually introduces a human bias.

VI. Conclusions

This paper has presented a novel approach of process mining, called the Map Miner Method (MMM). The main contribution of this work is to build from theory a method, which fully constructs the topology of an intentional process model (Map), only based on users' activity logs. Whereas process mining techniques focus how a process is enacted, MMM focuses on why a process is enacted and what a process must do. MMM helps understanding the users' intentions and strategies while enacting processes. This allows discovering actual process models and checking the conformance of the models. MMM requires only users' traces of activities (event logs) as inputs. This makes the method easily applicable to any dataset. In this paper, MMM was applied on a large-scale case study (developers of Eclipse UDC), which demonstrates its scalability. This paper focused on discovery of Map process model topology. In other words, the proposed method finds the relationships between activities to discover the strategies and where they lead, *i.e.* the intentions. However, the names of these strategies and intentions are still inferred semi-automatically. Indeed, the proposed method is able to extract automatically some topics related to each strategy. This establishes a preliminary base to infer manually the names of strategies and intentions. In the future, this procedure can be fully automated by building more sophisticated ontologies from these discovered topics. These ontologies should take into account the context in which the processes are enacted as well as the situation at hand. This makes the discovered Map more context-sensitive.

This paper mainly focused on discovery of intentional process models. However, the usefulness of the MMM is not only limited to process discovery. In the future the discovered process models can be useful for multiple concerns. For instance, at the project management level, it allows checking the alignment between prescribed process models and what stakeholders actually do; or at the application level, monitoring users and provides run-time recommendations. This helps improving the software usability by using anterior developers' activities as a guideline by assisting the novice or unfamiliar users learning system features by making the task of learning easier. For example, when users' intentions are known, they can be recommended which strategies and activities might be useful to fulfill their intentions. This guideline is adapted to the users' context taking into account the experiences of previous users and actual users' intentions. These phases can be automated and integrated as modules of MMM. Furthermore, the discovered Map enables users to be more efficient in their operations by adapting the system to the users' needs. Assisting users step by step using a Map increases their confidence and satisfaction in the enactment of a process. All these points contribute to improving the usability of the software products.

So far, the tools in process mining fields represent the process in terms of activities. To offer an intentional vision on processes, the implemented tool in this thesis will be plugged into ProM toolkit [55]. This allows the extraction of knowledge about a process from its process execution logs in an intentional manner.

References

[1] A. Rozinat, *Process Mining Conformance and Extension*. PhD thesis, Technische Universiteit Eindhoven, 2010.

[2] A. Rozinat, A. A. de Medeiros, C. W. Günther, A. Weijters, and W. M. van der Aalst, *Towards an evaluation framework for process mining algorithms*. Beta, Research School for Operations Management and Logistics, 2007.

[3] W. Van der Aalst, T. Weijters, and L. Maruster, "Workflow mining: Discovering process models from event logs," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 16, no. 9, pp. 1128–1142, 2004.

[4] W. M. Van der Aalst and W. van der Aalst, *Process mining: discovery, conformance and enhancement of business processes*. Springer, 2011.

[5] C. Rolland and C. Salinesi, "Modeling goals and reasoning with them," in *Engineering and Managing Software Requirements*, pp. 189–217, Springer, 2005.

[6] E. B. Swanson, "Management information systems: appreciation and involvement," *Management Science*, vol. 21, no. 2, pp. 178–188, 1974.

[7] B. Christie, "Face to file communication: A psychological approach to information systems," 1981.

[8] F. D. Davis, R. P. Bagozzi, and P. R. Warshaw, "User acceptance of computer technology: a comparison of two theoretical models," *Management science*, vol. 35, no. 8, pp. 982–1003, 1989.

[9] I. Ajzen and M. Fishbein, "Belief, attitude, intention, and behavior: An introduction to theory and research by martin fishbein; icek ajzen," 1975.

[10] L.-H. Thevenet and C. Salinesi, "Aligning is to organization's strategy: the instal method," in *Advanced Information Systems Engineering*, pp. 203–217, Springer, 2007.

[11] C. Rolland, P. Loucopoulos, V. Kavakli, S. Nurcan, *et al.*, "Intention based modelling of organisational change: an experience report," *Proceedings of Evaluation of Modeling Methods in Systems Analysis and Design*, 1999.

[12] C. Salinesi and C. Rolland, "Fitting business models to system functionality exploring the fitness relationship," in *Advanced Information Systems Engineering*, pp. 647–664, Springer, 2003.

[13] C. Rolland, "Modeling the requirements engineering process," in *Information Modelling and Knowledge Bases V: Principles and Formal Techniques: Results of the 3rd European-Japanese Seminar, Budapest, Hungary, May*, pp. 85–96, 1993.

[14] R. Deneckère and E. Kornyshova, "Process line configuration: An indicator-based guidance of the intentional model map," in *Enterprise, Business-Process and Information Systems Modeling*, pp. 327–339, Springer, 2010.

[15] C. Rolland, M. Kirsch-Pinheiro, and C. Souveyet, "An intentional approach to service engineering," *Services Computing, IEEE Transactions on*, vol. 3, no. 4, pp. 292–305, 2010.

[16] M. Jarke and K. Pohl, "Establishing visions in context: towards a model of requirements processes," in *ICIS*, pp. 23–34, 1993.

[17] S. Najar, M. Kirsch-Pinheiro, and C. Souveyet, "Towards semantic modeling of intentional pervasive information systems," in *Proceedings of the 6th International Workshop on Enhanced Web Service Technologies*, pp. 30–34, ACM, 2011.

[18] J. Ralyté, R. Deneckère, and C. Rolland, "Towards a generic model for situational method engineering," in *Advanced Information Systems Engineering*, pp. 95–110, Springer, 2003.

[19] I. Mirbel and J. Ralyté, "Situational method engineering: combining assembly-based and roadmap-driven approaches," *Requirements Engineering*, vol. 11, no. 1, pp. 58–78, 2006.

[20] C. Rolland, C. B. Achour, C. Cauvet, J. Ralyté, A. Sutcliffe, N. Maiden, M. Jarke, P. Haumer, K. Pohl, E. Dubois, *et al.*, "A proposal for a scenario classification framework," *Requirements Engineering*, vol. 3, no. 1, pp. 23–47, 1998.

[21] G. Khodabandelou, "Contextual recommendations using intention mining on process traces," in *Proceedings of 7th Intl. Conf. on RCIS*, 2013.

[22] G. Khodabandelou, C. Hug, R. Deneckère, and C. Salinesi, "Process mining versus intention mining," in *Enterprise, Business-Process and Information Systems Modeling*, pp. 466–480, Springer, 2013.

[23] G. Khodabandelou, C. Hug, R. Deneckere, C. Salinesi, *et al.*, "Supervised intentional process models discovery using hidden markov models," in *Proceedings of 7th Intl. Conf. on RCIS*, 2013.

[24] E. Yu, "Modelling strategic relationships for process reengineering," *Social Modeling for Requirements Engineering*, vol. 11, p. 2011, 2011.

[25] A. Dardenne, A. Van Lamsweerde, and S. Fickas, "Goal-directed requirements acquisition," *Science of computer programming*, vol. 20, no. 1, pp. 3–50, 1993.

[26] C. Rolland, N. Prakash, and A. Benjamen, "A multi-model view of process modelling," *Requirements Engineering*, vol. 4, no. 4, pp. 169–187, 1999.

[27] P. Soffer and C. Rolland, "Combining intention-oriented and state-based process modeling," in *Conceptual Modeling–ER 2005*, pp. 47–62, Springer, 2005.

[28] L. Rabiner and B. Juang, "An introduction to hidden markov models," *ASSP Magazine, IEEE*, vol. 3, no. 1, pp. 4–16, 1986.

[29] Eclipse, "Filtered udc data," 2013.

[30] Y. Bengio, "Learning deep architectures for ai," *Foundations and trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[31] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains," *The annals of mathematical statistics*, vol. 41, no. 1, pp. 164–171, 1970.

[32] K. P. Burnham and D. R. Anderson, *Model selection and multi-model inference: a practical information-theoretic approach*. Springer, 2002.

[33] C. Rolland, "A comprehensive view of process engineering," in *Advanced Information Systems Engineering*, pp. 1–24, Springer, 1998.

[34] C. Fernstrom and L. Ohlsson, "Integration needs in process enacted environments," in *Software Process, 1991. Proceedings. First International Conference on the*, pp. 142–158, IEEE, 1991.

[35] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[36] Eclipse, "Usage data collector." http://eclipse.org/org/usagedata/, 2013.

[37] J. E. Cook and A. L. Wolf, "Discovering models of software processes from event-based data," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 7, no. 3, pp. 215–249, 1998.

[38] D. Lorenzoli, L. Mariani, and M. Pezzè, "Automatic generation of software behavioral models," in *Software Engineering, 2008. ICSE'08. ACM/IEEE 30th International Conference on*, pp. 501–510, IEEE, 2008.

[39] R. Agrawal, D. Gunopulos, and F. Leymann, *Mining process models from workflow logs*. Springer, 1998.

[40] A. Datta, "Automating the discovery of as-is business process models: Probabilistic and algorithmic approaches," *Information Systems Research*, vol. 9, no. 3, pp. 275–301, 1998.

[41] B. F. van Dongen and W. M. van der Aalst, "Multi-phase process mining: Building instance graphs," in *Conceptual Modeling–ER 2004*, pp. 362–376, Springer, 2004.

[42] A. J. Weijters and W. M. van der Aalst, "Rediscovering workflow models from event-based data using little thumb," *Integrated Computer-Aided Engineering*, vol. 10, no. 2, pp. 151–162, 2003.

[43] J. Herbst, "A machine learning approach to workflow management," in *Machine Learning: ECML 2000*, pp. 183–194, Springer, 2000.

[44] W. M. Van Der Aalst, H. A. Reijers, and M. Song, "Discovering social networks from event logs," *Computer Supported Cooperative Work (CSCW)*, vol. 14, no. 6, pp. 549–593, 2005.

[45] M. Song and W. M. van der Aalst, "Towards comprehensive support for organizational mining," *Decision Support Systems*, vol. 46, no. 1, pp. 300–317, 2008.

[46] W. M. van der Aalst, B. F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A. Weijters, "Workflow mining: a survey of issues and approaches," *Data & knowledge engineering*, vol. 47, no. 2, pp. 237–267, 2003.

[47] J. E. Cook and A. L. Wolf, "Automating process discovery through event-data analysis," in *Software Engineering, 1995. ICSE 1995. 17th International Conference on*, pp. 73–73, IEEE, 1995.

[48] S. Das and M. C. Mozer, "A unified gradient-descent/clustering architecture for finite state machine induction," in *NIPS*, pp. 19–26, Morgan Kaufmann, 1994.

[49] A. W. Biermann and J. A. Feldman, "On the synthesis of finite-state machines from samples of their behavior," *Computers, IEEE Transactions on*, vol. 100, no. 6, pp. 592–597, 1972.

[50] L. E. Baum and T. Petrie, "Statistical inference for probabilistic functions of finite state markov chains," *The annals of mathematical statistics*, vol. 37, no. 6, pp. 1554–1563, 1966.

[51] J. E. Cook and A. L. Wolf, *Event-based detection of concurrency*, vol. 23. ACM, 1998.

[52] J. E. Cook and A. L. Wolf, "Software process validation: quantitatively measuring the correspondence of a process to a model," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 8, no. 2, pp. 147–176, 1999.

[53] A. A. De Medeiros and A. Weijters, "Genetic process mining," in *Applications and Theory of Petri Nets 2005, volume 3536 of Lecture Notes in Computer Science*, Citeseer, 2005.

[54] J. Herbst and D. Karagiannis, "Integrating machine learning and workflow management to support acquisition and adaptation of workflow models," in *Database and Expert Systems Applications, 1998. Proceedings. Ninth International Workshop on*, pp. 745–752, IEEE, 1998.

[55] EUT, "Prom." urlhttp://www.processmining.org/prom/start, Nov. 2013.