Process Mining for Knowledge-intensive Business Processes

Marian Benner-Wickner,
Tobias Brückmann, Volker Gruhn
paluno – The Ruhr Institute for Software Technology
University of Duisburg-Essen
Essen, Germany
{marian.benner, tobias.brueckmann,
volker.gruhn}@uni-due.de

Matthias Book
Faculty of Industrial Engineering, Mechanical Engineering and Computer Science
University of Iceland
Reykjavik, Iceland
book@hi.is

ABSTRACT
In recent years, investigating opportunities to support knowledge-intensive business processes has gained increasing momentum in the research community. Novel contributions that introduce paradigms addressing the need for process execution flexibility form an alternative to traditional workflow management approaches and are mostly subsumed under the concept of adaptive case management (ACM). However, many of these approaches omit mining any kind of knowledge about such processes. This is because there is a gap between process mining, which works well for structured processes, and ACM, which mainly focuses on information system support for task management and collaboration using heterogeneous data sources. In this paper, we strive to bridge this gap by introducing a method for mining knowledge-intensive processes. It is part of *agenda-driven case management,* an ACM approach that follows the idea of mining common execution patterns while a case manager handles a flexible agenda.

CCS Concepts
• Applied Computing ➡ Enterprise Computing • Information Systems ➡ Expert Systems

Keywords
Knowledge Discovery; Process Mining; Case Management; Recommender Systems.

1. MOTIVATION
In several domains such as healthcare and insurances, actors are executing flexible business processes with a high amount of knowledge-intensive tasks. Such actors, often called case managers, track e.g. the healing progress of patients in order to shorten the time to occupational reintegration and thus minimize therapy costs. The execution path is usually determined by events that are hard to predict. Also, the data needed to drive the process cannot be defined in advance. Supporting such processes is a challenge since most of the mature BPM methods need a process model in order to suggest a workflow for a given context. However, such models are not available in *adaptive case management (ACM)* because the application of control flow mining on knowledge-intensive processes results in very complex "spaghetti" models [2, 5] or models with a very high level of abstraction [14]. The BPM answer to this dilemma is to mine for different (e.g. organizational) perspectives or to use distinct algorithms like the Fuzzy Miner [10], adapting cartography terms (e.g. abstraction and aggregation) in order to hide infrequent behavior. However, these are no ideal options for actors driving knowledge-intensive tasks since they either do not provide control flow support, or do so only at a level that shows well-known procedures.

As a result, several researchers and practitioners are working on ACM approaches, looking for ways to augment or adapt mature BPM efforts for application to knowledge-intensive processes. One of these approaches is *agenda-driven case management (adCM).* Formerly introduced as open process clouds [3], it follows the notion of a case manager working with a flexible agenda to organize his tasks (Fig. 1). This is similar to the "plan" concept which has been introduced in 2014 as part of the CMMN standard [15]. While working with the agenda, case managers' execution trails are monitored with the aim to identify templates, i.e. patterns of behavior occurring in a common context. In the conceptual field of agenda-driven case management, a set of templates is the counterpart to a single process model that can be mined from structured processes in traditional BPM.

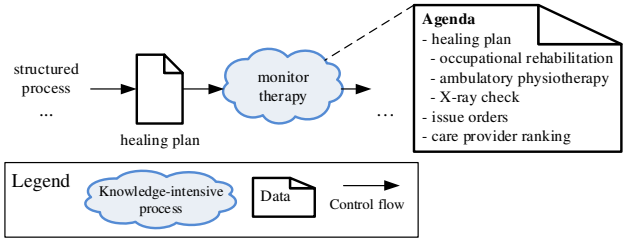

Fig. 1. Example agenda in rehabilitation management

In our previous work [4, 6], we already developed a tool to support this approach, and applied it to fields like the management of flexible software processes, thesis supervision and rehabilitation management. The tool so far focused mostly on the fundamental feature of providing an agenda, monitoring the execution paths, and some auxiliary functions such as annotation support and easy access to enterprise data. Our main goal however is to mine templates from execution trails, and use them to recommend courses of action in future cases.

ACM 978-1-4503-3721-2/15/10…$15.00
DOI: http://dx.doi.org/10.1145/2809563.2809580

The contribution of this paper is to introduce an approach for mining and recommending templates, and to provide a cross-validated evaluation based on a real-life event log. Section 2 will give a brief overview of our approach and its implementation in the tool. In Sect. 3, we then describe our setup for evaluating the feasibility and utility of the approach. The evaluation results are reported and discussed in Sect. 4, followed by a brief overview of the related work in Sect. 5. We close with an outlook on future enhancements in Sect. 6.

2. TEMPLATE DISCOVERY AND RECOMMENDATION

The main goal of agenda-driven case management is to support case managers during case execution. Besides helping them to access a broad variety of heterogeneous information sources and keep track of key activities, providing them with process knowledge from past cases is a key aspect of execution support. For certain case instances, the case manager might feel the need to adapt process knowledge from similar previous cases that is known to be applicable in the current context. Consider, for example, a case manager who is beginning to monitor a therapy. To structure what he has to do to handle the case, it would be useful to be inspired from what is often/usually done in the "monitor therapy" process, e.g. what kind of expertise is usually consulted in the healing steps, if medical datasets should be checked, and what further sources of knowledge should be considered.

In the field of structured processes, workflow management systems can predict future activities based on the context and the process model. Unfortunately, knowledge-intensive processes usually lack such a cohesive model and, moreover, the context has more dimensions than just the past completed activity. Hence, providing the case manager with modular building blocks of process knowledge seems more suitable than trying to identify one "master" process. Consider, for example, the agenda on the right of Fig. 1: While the top-level agenda items are generic and therefore suitable for common process knowledge, the sub-items of "healing plan" are very specific to the patient's condition. Since a case with the exact same set of conditions is very unlikely, these items are only suitable for common process knowledge when they are frequent enough.

For this purpose, the adCM approach relies on so-called *templates* containing common agenda items that can be identified by the mining algorithms introduced below. Templates have the same hierarchical structure as agendas and are stored in a knowledge base. As a result, they are accessible to recommender systems. Whenever such a system concludes that one or many templates match the current case context, it provides the templates to the case manager who can then decide whether or not to include (parts of) them into his current agenda. In the following subsections, we will introduce a template discovery algorithm named "adCM Miner" and show how the discovered templates can be recommended during case execution.

2.1 Template Discovery using adCM Miner

Imagine an agenda item "1st outpatient consultation" that has often been associated with the parent item "radiotherapy". Even though the discovery algorithm cannot understand the semantics of the item labels (e.g. that such a consultation is always the first step in the radiotherapy group), the domain experts driving the process do. We therefore argue that templates can be identified by comparing completed agendas of past cases and identifying frequent patterns within these agendas.

However, identifying frequent agenda items is not that easy since case managers can label the items freely. This creates a lot of noise in the data as the same activity may be labeled quite differently in different cases. As a result, comparing past instances is much more complicated than in traditional process mining, where event logs contain a sound nomenclature as stated in the fourth guiding principle of the Process Mining Manifesto [12]. To deal with this, the adCM Miner starts with two preprocessing steps: It first clusters the agenda item labels from finished agendas found in the case file archive and then replaces the original labels by the cluster IDs (see Fig. 2).

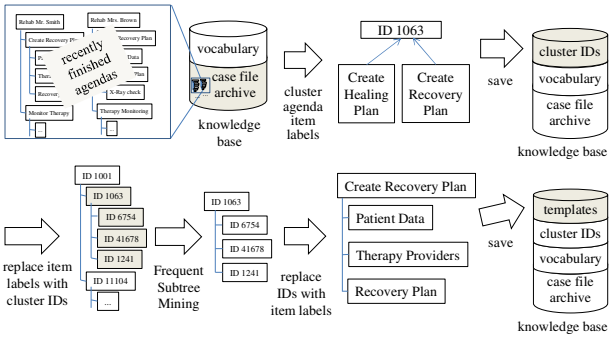

Fig. 2. Main processing steps performed by the adCM Miner

To perform the clustering, the algorithm aggregates activity labels into a set of synonyms (aka "synsets") originating from a controlled vocabulary or better yet, a domain ontology containing knowledge about concept relationships (e.g. ICD-10). This way, the algorithm is able to find matches based on synonyms that would otherwise remain undiscovered. Given that the synsets are updated whenever the case mangers' concepts change, the algorithm can also address the challenge of concept drift [2]. Besides these semantic concerns, clustering also has a technical advantage: Since it is much faster to compare numerical values than text, replacing the agenda item labels by cluster IDs heavily reduces the runtime complexity of the subsequent mining step. However, before recommending mined templates, the IDs have to be transformed back into human-readable labels. If there are multiple synonyms associated with an ID, the algorithm needs to decide which one fits best. We believe that it would be impractical to ask always case managers to decide between the terms. Instead, we propose to use the most frequent term associated with the same ID. The clustering algorithm is therefore designed to count the frequency of each synonym during cluster analysis. Obviously though, this ability to mine templates without an in-depth understanding of their semantics comes at the cost of expressiveness.

The above-mentioned preprocessing steps are followed by the core of the overall discovery algorithm. It is called "frequent subtree mining" since it is designed to identify frequent parts of agendas, which are subtrees of the overall agenda hierarchy. Hence, we can apply tree mining, a sub-discipline of data mining. Originally developed to match different XML files, there are various tree mining algorithms that can be used to find matches between hierarchical data structures [9, 23, 24]. Since these algorithms are classified by the context of usage, we can identify the most suitable mining algorithm by choosing the right characteristic from each of the following attributes [11]:

1. Input type: the type of trees the algorithm operates on (e.g. do the trees have a root, and are they ordered or unordered?)
2. Modus operandi: the way the algorithm counts frequent occurrences (e.g. counting each occurrence, or just noting if a subtree occurs multiple times within a tree)
3. Output type: the type of mined subtrees (e.g. *induced subtrees* reflect only direct parent/child relationships while *embedded subtrees* also reflect ancestor/descendant relationships across nodes)

When discovering templates, the input type is a "forest" of agenda trees. By definition, these trees do have a root, and since the case manager is free to structure the agenda in any way, its nodes are not ordered. The adCM Miner therefore has to cope with rooted, unordered trees. Concerning the modus operandi, there is no reason why the algorithm should count a subtree only once even when it occurs multiple times in an agenda, so we looked for algorithms that count each subtree occurrence. Finally, choosing the right output type requires a bit more consideration: Recall that the algorithm outputs templates mined from finished agendas. These agendas are in turn structured freely by the case manager, and their hierarchy may represent sophisticated semantics. When mining embedded subtrees, it would be possible to omit levels of that hierarchy, corrupting the underlying semantics. We therefore need to choose an algorithm that produces induced subtrees.

After comparing these characteristics with the algorithms described in literature, we chose the CMTreeMiner developed by Chi et al. [9]. It is capable to work on rooted unordered trees and produces induced subtrees. In contrast to most other algorithms, it does not calculate the frequency of every possible subtree, which is of course expensive. Instead, it only keeps those above a given frequency threshold ("minimum frequency"). This threshold is a relative measure calculated by counting the amount of agendas that contain the subtree. Since we do not need the results for less-frequent subtrees, we can use this very efficient algorithm.

In brief, the CMTreeMiner algorithm works as follows: Beginning with the most frequent root nodes in the forest of agenda trees, it continues to build subtrees by adding child nodes. Note that adding nodes usually decreases the probability that the whole subtree can be found in many agenda trees. So after adding each child, the algorithm has to recalculate the frequency of the whole subtree by counting equivalent subtrees in the agenda forest. If the frequency is still above the threshold, the new child node remains in the subtree. If the frequency drops below the threshold, the node gets replaced by the next node. This way, the algorithm produces trees that are maximal, i.e. it is impossible that there is an undiscovered supertree (i.e. a tree with additional nodes) having an equal or even higher frequency.

The algorithm has two advantages: Firstly, pruning all search paths that cannot result in maximal trees allows for faster computations. Secondly, the algorithm can prune uninteresting results and improve mining quality [18]. That is, candidates that can be subsumed by others are not interesting. For example, a frequent agenda item "Create healing plan" is not interesting if there is also a frequent combination with a child agenda item (e.g. "Discuss diagnosis"). However, it has the slight disadvantage that in its original form, it counts multiple occurrences of a subtree only once in a tree. We therefore adapted the algorithm so that it counts each occurrence.

Due to space limitations, we kindly refer to the work of Chi et al. for detailed information about the internal functioning of the CMTreeMiner algorithm [9].

2.2 Recommender System

Operational support during process execution is divided into three sub-functions: detect, predict and recommend [2]. The key goal of the recommender function is to make a suggestion for the next process steps. In the field of well-known business processes, traditional workflow management systems provide such process knowledge using a fixed, predefined, and ordered collection of activities defined in a process model. However, in agenda-driven case management, the case manager is supported by a flexible agenda and a set of templates instead of a fixed process model. We therefore designed and implemented a system that recommends templates while the case manager interacts with the agenda.

Of course, a recommender system that actively suggests templates could be perceived as intrusive if it disturbs case managers during knowledge-intensive activities. On the other hand, if case managers have to explicitly ask a more passive recommender system for suggestions, they may find this tedious and miss valuable recommendations at times. Our recommender system therefore tries to strike a balance between active suggestions and passive execution support. It is sensitive to the current context (the agenda) and provides recommendations just in time when they are assumed to be suitable and not disturbing the case manager. To achieve this, we designed a certainty-aware recommendation procedure providing three-tiered support:

Level 0. Initially, the agenda is blank or contains just a few items. At this stage, it is too small to provide sufficient context information for identifying any matching template. As a consequence, on this level the recommender system is only in observation mode and does not affect the case manager's behavior.

Level 1. The next level is entered when the agenda is big enough to allow suggestion of a template. Based on our experience, we use a minimum of three agenda items as a default threshold. However, a more precise recommendation can only be made based on experience in the field. While this agenda size is enough for the recommender system to find templates, it is still quite small, so the recommender system cannot be certain yet which of the found templates fit best. We therefore do not to suggest them to case managers right-away – instead, we designed the recommender system to create a shortlist of template candidates matching the current agenda. It then provides just the template item labels as auto-complete suggestions whenever the case manager is about to add a new agenda item. This strikes a balance between comprehensive support and an unobtrusive user experience: The case manager does not need to explicitly accept or reject suggested templates, but if he uses one of the suggested auto-complete items rather than just typing over them, the recommender system can be more certain which of the matching templates might be of interest to the case manager. At level 1, the recommender system thus monitors if the state of the agenda converges, with the help of the auto-complete function, to a subset of the matching templates. That is, for each added agenda item that is also contained in some of the matching templates, the system is more certain which templates should be actually recommended. If the recommender system is certain enough to provide this subset of templates, it proceeds to level 2. This certainty threshold needs to be calibrated according to the very specific support needs of the case manager: It should be set higher (e.g. 6-8 items) if the suggested templates are not precise enough, i.e. they contain many superfluous items not covered by future agenda items, and it should be set lower (e.g. 2-4 items) if the

support level is not sufficient, i.e. if the agenda already contains most of the recommended template items.

Level 2. Once the recommender system has become sufficiently certain that a certain subset of matching templates would be suitable in the current context, it actively suggests them to the case manager. The case manager can then choose a template and decide if he wants to import it as a whole or just those parts he considers relevant.

We believe this three-level approach is more convenient than traditional approaches since it reduces unsolicited recommendations and reflects that the case manager is in charge, not the operational support system. More information on the underlying system architecture, implementation details and additional mining and support perspectives can be found in our previous discussion of the adCM prototype [6] and in [19].

3. EVALUATION DESIGN AND SETUP

Our goals for evaluating the adCM template mining components are twofold: Firstly, we want to show that it is feasible to generate templates based on agendas from past cases, and that matching templates can be identified and recommended during execution of a new case. Secondly, we want to demonstrate the approach´s utility by comparing the results with the performance of process mining algorithms.

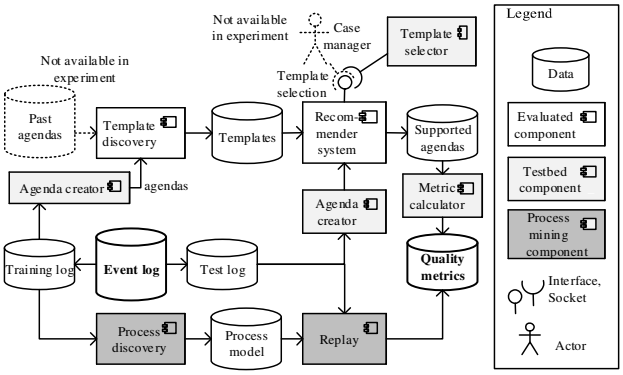

Fig. 3. Data flow for the adCM (top) and process mining (bottom) toolchain, both starting with the event log and ending with the quality metrics

Obviously, conducting a comparative study to evaluate these criteria is very difficult, since this would require an organization that is willing to handle its cases using two different approaches, ideally with two separate but comparable groups of case managers. Therefore, our evaluation is designed as an "offline" experiment based on a real-life process event log instead: As Fig. 3 shows, we run this event log both through our template discovery and recommender components (top path) as well (for comparison) through a common process mining toolchain (bottom). Since we neither have historical agendas nor human case managers available in this automated experiment, we simulate them through components of our testbed, as described in more detail below. We finally calculate quality metrics for the results of both paths, which we use to compare the two approaches' relative feasibility and utility. In the following subsections, we will explain in seven steps how the evaluation was set up and executed.

3.1 Step 1: Select quality metrics

In this step, we chose metrics for measuring the utility of the mining and recommender components. For the purpose of reproducibility, we looked for existing metrics to adapt, rather than defining new ones. The quality metrics discussed most prominently in process mining literature are fitness, precision, generalization and simplicity [2, 8].

Precision measures how a model allows for too much behavior. For example, if every model element is observed (replayed) by a given log, precision is high. If there are many elements in the model that are not observed, the precision is low. Precision can detect symptoms like a "flower model" which is simple, fits and generalizes perfectly, but allows for almost every behavior [2]. From the case manager's point of view, precision can be interpreted as the efficiency of the model in supporting his tasks. If many activities are recommended that do not actually fit to future tasks, the model is not precise, which may cause distractions during case execution. Since our templates do not reflect a control flow, we could not adapt the definition of precision introduced by Buijs et al [8]. Instead, we calculated it by relating the number of actually replayed model elements to the total model elements. Or, in terms of agenda-driven case management, we related the number of replayed template items to the total number of template items imported by the recommender.

Fitness (aka recall) measures how much of the behavior in the training log is reflected by the model. For example, if there are many events in the log that are not described by the model, fitness is low. If the model only reflects the "happy path" of a process containing just the most frequent activities, it may be simple and even precise. However, it is not fitting every trace. From the case manager's point of view, fitness is important because traces of knowledge-intensive processes are naturally much more diverse compared to those of structured processes, so even the most frequent behavior would only be covered by very few traces. As a result, optimizing other measures at the cost of fitness would harm operational support. We measured fitness in a very similar way to the definition of Buijs et al. [8] by relating the number of actually replayed events to the total events.

Generalization indicates how a model can abstract from the behavior of the log it has been mined from, in order to also reflect previously unknown behavior in upcoming traces. For example, an "enumeration model" containing distinct paths for each trace perfectly fits the training log [2]. However, it does not generalize well. On the opposite of the spectrum, the flower model perfectly generalizes the behavior, as explained above, but it does so at the cost of precision. Templates that generalize well are important for a case manager because of the nature of knowledge-intensive tasks where corresponding logs are never complete. As a result, no training log is complete enough to ensure that good fitness also leads to good generalization. Buijs et al. introduce a metric for generalization that is based on the assumption that infrequently replayed model elements indicate bad generalization [8]. While this approach makes sense in the domain of structured processes, things are different in case management: Consider, for example, a successful therapy for a rare disease. Of course this is less frequent behavior, but one should be wary of truncating the corresponding model elements just because of their low frequency. We therefore argue to measure generalization closer to its original definition: If the model fits in the same way to the training log and a log that is different from it, then generalization is high; if it only fits the training log, generalization is low. So we related the fitness of the test log to the fitness of the training log.

Simplicity follows the idea that the simplest model able to explain some behavior is the best model. It is obvious that simplicity is in conflict with fitness or precision on the one hand and benefits from generalization on the other hand. For example, fitness and precision are at maximum in the enumeration model. However, this model is huge as it contains every trace. On the opposite side, the simplest model is probably the flower model, which however lacks precision. Truncating less frequent behavior leads to a simple model as well, which however lacks fitness. Due to the very different concepts of a single heavy-weight process model in traditional process mining, and a set of lightweight templates in agenda-driven case management, the simplicity measures introduced by Bujis et al. could not be straightforwardly adapted for our approach. However, we believe that the size of the model most likely corresponds to the size of the total number of all template items. We therefore measured simplicity by relating the total number of mined template items to the total number of log events.

3.2 Step 2: Select an event log

Choosing a suitable event log is important since every subsequent step is based on this decision. First, the process that produced the log must be knowledge-intesive. Second, the process' dataset should be publically available in order to address the need for representative benchmarks [12].

The IEEE task force on process mining provides several event logs to the research community. They have been mined from real-life processes using common tools. Two of these event logs are identified as "ad-hoc structured", which mostly fits our notion of knowledge-intensive business processes. However, one log containing activities from a road traffic fine management process has just been added shortly before the submission deadline, so we were not able to use it for the evaluation. Hence, we chose the real-life event log from a Dutch academic hospital. It contains 1143 traces with more than 15k events (i.e. process activities).

3.3 Step 3: Prepare for cross-validation

Recall that it is theoretically trivial to mine for a model with perfect fitness, although it may not be simple or precise. Hence, we applied cross-validation in order to measure how the model, in terms of all mined templates, can generalize. To achieve this, we divided the data set according to common data mining practice [2, 7] into a training data set and a test data set. The training data set is used to generate the model (i.e. templates and process models, respectively). The test data set is used to replay the model (using the recommender system and process mining replay algorithms, respectively) and to calculate the quality metrics discussed above.

Since there is no common rule for a split ratio, we initially simply split the log in half. In a trial run of our evaluation, we however found that the first half of the traces in the Dutch hospital log is, on average, more diverse than the second half. To avoid biased results, we therefore split the log randomly in our actual evaluation run by randomly assigning each trace of the original log either to the training data set (which ended up containing 574 traces) or the test data set (which ended up at 569 traces).

3.4 Step 4: Select process mining algorithms

To follow the idea of representative process mining benchmarks [12], we need a baseline to compete with. We therefore conducted the experiment as similarly as possible to related approaches. To choose appropriate "traditional" discovery algorithms from the related work that our adCM algorithms can compete with, we first needed to define our requirements: Knowledge-intensive processes, like the one recorded in the selected event log, contain infrequent behavior to a large extent. In process mining terms, this is called "noise" [2]. We should therefore use discovery algorithms that are designed to cope with this. Additionally, the algorithms should be available in the ProM toolkit so that they are accessible and executable on the standardized event log format XES. Based on these requirements, we selected five discovery algorithms for our baseline: the ILP Miner [20], the InductiveMiner [13], the Heuristics Miner [22], the Flexible Heuristics Miner [21] and the ETM algorithm [8].

3.5 Step 5: Set up agenda creator

Recall that template discovery needs a case file archive containing past agendas instead of a classical event log. The main difference between these is that activities in an agenda are structured in a tree, not a sequence. To reflect this in our experimental setup, we had to find a way to transform the traces in the event log into agendas that could previously have been executed by case managers. For this purpose, we implemented an agenda creator component generating an agenda for each trace in the event log. Obviously, there are many strategies a case manager might follow to create an agenda (e.g. sorting the individual activities along some pivotal milestones, concepts or deliverables). And of course, the resulting templates look differently depending on which agenda creation strategy has been chosen.

To simulate this, we performed trial runs before our actual evaluation, in which we ran the whole experiment several times with different simulated agenda creation strategies, in order to examine their impact on the observed quality metrics. In order to find possible strategies, we examined the event metadata for promising attributes that could represent parent agenda items. Promising attributes should provide a fairly reasonable semantic and be suitable for clustering the activities into groups of rather similar size. A deep inspection of the log showed that the attributes "org:group" and "specialism code" can cluster the activities in groups of fairly similar size. They can also reflect semantic relationships – for example, "org:group" indicates the organizational entity in which the activity has been executed, which would indeed be a reasonable parent agenda item.

Our trial runs showed that the choice of agenda creation strategy did not really affect the resulting quality metrics: The values only differed by 0.27% on average with a standard deviation of 17.66% (which is actually not surprising since the template discovery algorithm just mines frequent subtrees, irrespectively of how the trees are created). For our actual evaluation, we therefore created the agendas based on the organizational classification strategy as we believe it is based on the most realistic parameters.

3.6 Step 6: Set up template selector

Since our evaluation is based on an offline simulation, we need a component that replaces the case manager who would ordinarily select one of the recommended templates (i.e. a component that simulates the case managers' choices on level 2 of the three-tiered recommender system described in Sect. 2.2). To do this, the component needs a strategy for deciding which of the matching templates should be applied to the current agenda. Possible strategies could be to select the template with the most matching items in the current agenda (raising precision) or to select the largest template (raising fitness). We implemented a hybrid strategy that strives to balance fitness and precision by selecting the largest template containing a minimum number of matching items.

Obviously, the choice of strategy directly affects the resulting agenda. However, these strategies are extremely simple compared

to the reasoning a case manager would employ in choosing from the recommendations, based on his qualification and years of experience. With regard to the quality metrics that we are finally going to calculate from the resulting agendas, we therefore assume that our automated template selection strategy will only represent a lower bound on the overall system performance – if a human case manager picked from the recommended templates, the resulting quality metrics should turn out even better.

3.7 Step 7: Perform and discuss evaluation

We conducted the evaluation in two steps: We first ran the adCM toolchain containing the template discovery and the recommender component, and then performed the traditional process mining steps (discovery and replay) on the same original event log for comparison. As shown in Fig. 3, both experiments follow the cross-validation approach and use the same dataset.

We started the adCM toolchain by transforming the event log into a forest of agendas and imported them into a graph database for the purpose of persistency. To find out how the minimum frequency threshold selection of the adCM miner affects the overall quality, we then ran the adCM miner using different thresholds (see the next section for detailed results). We continued by "replaying" the agendas using the recommender system and the mined templates. This step has been repeated for each sample threshold selected above and is conducted as follows: After loading the XES log, a blank agenda is created for each trace in the event log and the system is in level 0. Then each event of the trace is transformed into an agenda item and imported into the agenda, proceeding to level 1. If such an item already exists, the agenda item has been denoted as "replayed", which is important for metrics measurement. During adding/replaying the event, the recommender system checks whether templates can be applied to the current agenda state. If so, the system proceeds to level 2 and a template is applied by the selector. After replaying the test log, we finished the adCM toolchain by calculating the metrics from the template-supported agendas. We then proceeded with the comparison experiment by feeding the original event log into the process mining toolchain, applying different discovery algorithms. The results of both approaches are discussed in the next section**.**

4. RESULTS AND DISCUSSION

In this section, we will present the discovery and replay results. In either case, we start with the results from the selected baseline process mining algorithms, and proceed with the adCM Miner.

4.1 Discovery

Unfortunately, not every traditional process discovery algorithm could cope with the complexity of the event log. For example, the ILP algorithm and the ETM algorithm reproducibly failed after days of CPU processing due to excessive memory consumption – even using several different parameters addressing performance issues. These algorithms are therefore not considered in the discussion below. The only algorithms that successfully produced a process model are the InductiveMiner, the Heuristics Miner and the Flexible Heuristics Miner, each of which produced models containing more than 600 nodes.

To test the feasibility of our approach, we first need to check if the adCM Miner is able to discover templates from a real-life event log. As explained in Sect. 3, we therefore applied our mining algorithm using different minimum frequency thresholds. The results are shown in Fig. 4.

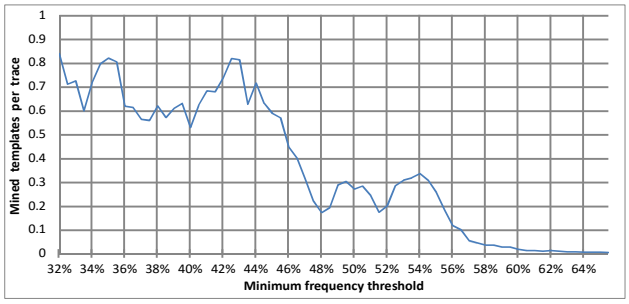

Fig. 4. Effects of the minimum frequency threshold on the number of mined templates

As illustrated in the graph, the relationship between the minimum frequency and the number of templates is similar to quality vs. quantity: Raising the required minimum frequency for each template (denoted on the horizontal axis) in general reduces the number of templates found (denoted on the vertical axis on a normalized scale). However, the relation is not simply monotonically decreasing. This can be explained by the design of the mining algorithm: For example, it tries to find maximal trees and does not impose an upper limit on the template size. Therefore, the minimum frequency can sometimes only be achieved by many smaller templates, which are easier to find than larger templates, yielding a higher number of mined templates.

Before proceeding to the next step, we had to choose thresholds that promised to yield the most interesting insights during replay. This is an important decision since we expect that recommending templates with a high minimum frequency also affects fitness and precision. We therefore identified three "points of interest": The first and the last are local maxima in the amount of mined templates that are followed by significant drops for higher frequency values. This happens at a minimum frequency value of 42.5% and again at 54%. To investigate how the adCM Miner performs with a very small set of templates, we also selected one of the local minima at 51.5%.

4.2 Replay

Recall that among the five traditional process mining algorithms we choose as a baseline, only the InductiveMiner, Heuristics Miner and Flexible Heuristics Miner were capable to mine a process model from the event log. Unfortunately, the traditional algorithms struggle during replay, too: The algorithm replaying the models generated by the Flexible Heuristics Miner only produced results denoted as "unreliable". Also, there are no algorithms available in the ProM toolkit to replay process models produced by the Heuristics Miner. However, we performed a workaround by transforming the model into a Petri net, which can be replayed. After all this, we could at least replay a Petri net using the InductiveMiner, and a heuristics net transformed into a Petri net. The InductiveMiner and Heuristics Miner thus are the only algorithms that are able to mine control flow process models of knowledge-intensive processes that are replayable by the ProM toolkit without noise filtering or other preprocessing steps.

In order to show whether the adCM Miner is feasible and whether its discovered templates are actually utilizable by a case manager, we replayed the test log and calculated the quality measures. Table 1 shows a comparison of the metrics depending on the three different frequency thresholds selected in the previous subsection. It also shows the total amount of template items mined during play-in (used for calculating simplicity).

Table 1. Average quality metrics from all 569 traces of the test log (77,238 events).

Mining Algorithm	*Frequ. (%)*	*#Items*	*Pre. (%)*	*Fit. (%)*	*Gen. (%)*	*Sim. (%)*
adCM	42.50	7,909	61.56	30.70	92.21	89.76
	51.50	1,146	69.12	26.85	99.29	98.52
	54.00	4,816	**71.48**	21.39	99.65	93.76
Inductive	N/A	621	6.43	**100.00**	**100.00**	**99.19**
Heuristics*	N/A	980	33.33	56.20	N/A	98.73

*) Output had to be transformed into a Petri net for replay.

4.2.1 Precision and Fitness

The precision of the adCM Miner is measured at about 62–71%, so most of the suggested template items have actually been observed in future events. This indicates that our approach is capable of discovering useful templates. The table also indicates that when recommending high-frequency templates, the precision rises due to the minimum frequency threshold. Additionally, note that the template selector included in the recommender system is rather simple compared to the case manager. So it is safe to assume that the precision will get even better if a real case manager chooses the templates according to experience and insight into a case's specifics. These results are very promising compared to the process mining algorithms.

Although a high precision is necessary for unobtrusive support, it often comes at the cost of fitness. We therefore have to accept that just about 21–31% of the case manager's activities are covered by suggested template items. Admittedly, this is poor against the perfect fitness of the petri net mined by the Inductive Miner. However, we believe that a perfect model with a very bad precision (~6%) is inacceptable for support as the user would continuously have to decide between a huge number of alternative paths.

4.2.2 Generalization

All algorithms generalize very well throughout. That is, much of the fitness remains when using a different data set (e.g. the test log). So the template concept (and its mining algorithm, respectively) seems to be capable of abstracting from case-specific behavior just as the traditional process models do. However, we were not able to measure the generalization of the Petri net transformed from the Heuristics Miner model, because the replay algorithms produced errors or did not terminate. Apart from this, we can also observe that generalization somewhat depends on the minimum frequency. Apparently, it is easier to generalize behavior when the minimum frequency is high.

4.2.3 Simplicity

Depending on the frequency threshold, the adCM Miner can discover hundreds of templates, which total up to 7,900 template items. The resulting template set thus is obviously not human-readable. However, simplicity is a measure relative to the behavior that has to be explained by the model. So considering the fact that the test log contains 77,238 events, the model is relatively simple (especially the smaller set of templates having a minimum frequency of 51.50%). In other words, it has a good compression by means of the minimum description length.

Though the baseline process models are also not human-readable, they at least contain fewer elements and are therefore a little simpler. However, since the adCM algorithms only recommend single templates to the case manager, the simplicity of the underlying model is not as important as fitness and precision.

5. Related Work

Mining knowledge-intensive business processes as introduced in this paper bridges a gap between two main disciplines: process mining and ACM. The former aims at support for structured processes by mining process models that help actors driving these processes [2]. The latter aims at knowledge-intensive processes by providing architectures and paradigms for case management systems [17]. Both matured in the recent years with valuable contributions in theory (e.g. petri nets [20]) and practice (e.g. application domains [14]). In the process mining community there are also efforts to broaden the scope to less-structured processes like the case handling paradigm [1] or the algorithms referenced in this paper. However, as our experiments showed, they still rely on preprocessing like noise filtering before they are applicable to knowledge-intensive processes. Other approaches try to support such processes by mining alternative perspectives like social nets, circumventing the control flow mining issues. ACM approaches, on the other hand, usually have a data-centric perspective unable to detect execution trails like the approach proposed above.

The adCM Miner presented in this paper is also strongly related to the user task detection based on UICO [16]. However, the adCM approach directly monitors the process execution on task level, instead of input device interaction level. This way, there is no need to aggregate low-level interaction events to tasks.

6. Conclusion and Outlook

Since process mining algorithms are not primarily designed for knowledge-intensive processes, most of them fail at mining a model that can support ad-hoc activities, at least not without truncating infrequent behavior. We therefore introduced the template discovery and recommending techniques of agenda-driven case management. As an alternative to existing algorithms, template discovery is designed to enable process owners to extract process knowledge from past cases in terms of templates containing frequent activity patterns. This knowledge is then used by an integrated multi-tiered recommender component aiming at unobtrusive suggestions for case managers. In our experiments assessing the feasibility and utility of both components, 21-30% of the activities have been foreseen by the model. In other words, the approach cannot predict the majority of events. However, given that the course of events in knowledge-intensive processes is naturally hard to predict, we are content with that result. Moreover, 61-71% of the recommended activities actually occurred in future events, which shows that the system is quite efficient and does not "spam" the case manager with vague suggestions. Compared to other approaches, we deem the results quite promising, because the few models that could have been discovered by the process mining algorithms had a very poor precision. In our view, precision is paramount however since knowledge-intensive processes usually contain demanding tasks vulnerable to distraction.

In our future research, we plan to both improve the approach and keep evaluating its utility in practice: Firstly, we want to improve the quality of the matching algorithm by reflecting the structure of the agenda. That is, we will not only check the pure existence of a template item within an agenda, but also see if it exists in the same hierarchy. Secondly, we are quite convinced that the recommender should also take the individual support of each

template into account (e.g. to prioritize the matching templates). Finally, we plan to expand our experiments on further real-life event logs that are available for research as well as replacing the artificial template selection by a real case manager in order to measure the expected improvements in fitness and precision.

7. ACKNOWLEDGMENTS

We would like to thank Pascal Groß and Alexander Kalinowski for their contributions to implementing the adCM tool.

REFERENCES

[1] Aalst, W. M. van der, Weske, M., and Grünbauer, D. 2005. Case handling: a new paradigm for business process support. *DKE* 53, 2, 129–162.

[2] Aalst, W. van der. 2011. *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer Verlag.

[3] Benner, M., Book, M., Brückmann, T., Gruhn, V., Richter, T., and Seyhan, S. 2012. Managing and Tracing the Traversal of Process Clouds with Templates, Agendas and Artifacts. In *BPM 2012*.

[4] Benner-Wickner, M., Book, M., Brückmann, T., and Gruhn, V. 2013. Cloud Storage of Artifact Annotations to Support Case Managers in Knowledge-Intensive Business Processes. In *WISE 2012*. LNCS. Springer, Berlin, Heidelberg, 92–104.

[5] Benner-Wickner, M., Book, M., Brückmann, T., and Gruhn, V. 2014. Examining Case Management Demand using Event Log Complexity Metrics. In *2014 EDOC Workshops (EDOCW)*.

[6] Benner-Wickner, M., Book, M., Brückmann, T., and Gruhn, V. 2014. Execution Support for Agenda-Driven Case Management. In *ACM SAC '14*.

[7] Berry, Michael J. A. and Linoff, G. 2000. *Mastering data mining. The art and science of customer relationship management*. Wiley, New York.

[8] Buijs, J. C. A. M., Dongen, B. F., and Aalst, W. M. P. van der. 2012. On the Role of Fitness, Precision, Generalization and Simplicity in Process Discovery. In *OTM 2012*. LNCS. Springer, Berlin, Heidelberg, 305–322.

[9] Chi, Y., Yang, Y., Xia, Y., and Muntz, R. R. 2004. CMTreeMiner: Mining Both Closed and Maximal Frequent Subtrees. In *Advances in Knowledge Discovery and Data Mining*. LNCS. Springer, Berlin, Heidelberg, 63–73.

[10] Günther, C. W. and van der Aalst,Wil M. P. 2007. Fuzzy Mining – Adaptive Process Simplification Based on Multi-perspective Metrics. In *BPM 2007*. LNCS. Springer, Berlin, Heidelberg, 328–343.

[11] Hadzic, F., Tan, H., and Dillon, T. S. 2010. *Mining of Data with Complex Structures*. Studies in Computational Intelligence 333. Springer, Berlin, Heidelberg.

[12] IEEE Process Mining Task Force. 2012. Process Mining Manifesto. In *BPM Workshops 2012*. LNCS. Springer, Berlin, Heidelberg, 169–194.

[13] Leemans, S. J. J., Fahland, D., and van der Aalst,Wil M. P. 2013. Discovering Block-Structured Process Models from Event Logs - A Constructive Approach. In *PETRI NETS 2013*. LNCS. Springer, Berlin, Heidelberg, 311–329.

[14] Mans, R. S., Schonenberg, M. H., Song, M., van der Aalst,W. M. P., and Bakker, P. J. M. 2009. Application of Process Mining in Healthcare – A Case Study in a Dutch Hospital. In *BIOSTEC 2009*. Springer, Berlin, Heidelberg, 425–438.

[15] Object Management Group. 2014. *Case Management Model and Notation (CMMN),* formal/2014-05-05. http://www.omg.org/spec/CMMN/1.0/.

[16] Rath, A. S. 2010. *User Interaction Context - Studying and Enhancing Automatic User Task Detection on the Computer Desktop via an Ontology-based User Interaction Context Model,* Graz University of Technology.

[17] Swenson, K. D., Ed. 2010. *Mastering the unpredictable*. Meghan-Kiffer Press.

[18] Tagarelli, A., Ed. 2012. *XML Data Mining*. IGI Global.

[19] Terziev, Y., Benner-Wickner, M., Brückmann, T., and Gruhn, V. 2015. Ontology-Based Recommender System for Information Support in Knowledge-Intensive Processes. In *15th International Conference on Knowledge Technologies and Data-driven Business*. i-KNOW '15. ACM.

[20] van der Werf,J. M. E. M., van Dongen, B. F., Hurkens, C. A. J., and Serebrenik, A. 2008. Process Discovery Using Integer Linear Programming. In *PETRI NETS 2008*. LNCS. Springer, Berlin, Heidelberg, 368–387.

[21] Weijters, A. and Ribeiro, J. Flexible Heuristics Miner (FHM). In *SSCI 2011*, 310–317.

[22] Weijters, A. J. M. M. and De Medeiros, A. K. A. 2006. *Process Mining with the HeuristicsMiner Algorithm.* BETA Working Paper Series, WP 166.

[23] Yao, J. T. and Zhang, M. A Fast Tree Pattern Matching Algorithm for XML Query. In *IEEE ACM WI 2004*, 235–241.

[24] Zaki, M. J. 2004. Efficiently Mining Frequent Embedded Unordered Trees. *Fundam. Inf.* 66, 1-2, 33–52.