Supporting Collaborative Work by Learning Process Models and Patterns from Cases

Stefan Schönig, Michael Zeising, Stefan Jablonski
University of Bayreuth
Bayreuth, Germany
{stefan.schoenig, michael.zeising, stefan.jabonski}@uni-bayreuth.de

Abstract — **Recent work shows an increasing interest of the Business Process Management (BPM) community in unstructured, so-called "human-centric" processes. Case Management (CM) is a new trend that focuses on the support of collaborative human-centric processes. Although CM provides concepts that support human-centric work, processes have to be modelled beforehand in order to be supported by IT systems. Hence, a problem that arises when applying CM is that when organisations begin to formalize CM practice, it is often difficult to express rules controlling the applicability of tasks. Furthermore, fundamental complexity challenges arise when applying CM in practice. In this contribution, we provide a solution to these two issues. We propose that managing human-centric processes should start with model skeletons that serve as a lattice where initial process execution can lean against. Additionally, by tracking different process cases, substantial process knowledge is recorded. Exploring process history might reveal certain recurring patterns that serve as dynamic guidance enhancement for CM systems. In this way, process models might evolve over time, become more and more complete and better reflect operational reality.**

Keywords — Business Process Management, Process Mining, Adaptive Case Management, Association Rule Mining, Process Observation

I. Introduction

During the last decade, Business Process Management (BPM) has established itself as the traditional approach to increase business productivity [1]. By providing tools that enable companies to define and map their processes to an IT environment, conventional BPM primarily aims at supporting "routine" work and therefore automating highly repetitive and structured processes. Thereby, it is mandatory for conventional BPM that processes have to be modelled completely before they can be interpreted and executed. Thus, these approaches are more suitable for highly predictable routine processes [2]. As indicated by many authors, conventional Workflow Management Systems (WfMS) are too restrictive [3]. In conventional WfMS, the only way to handle exceptions is to go behind the system's back, i.e., to bypass the system. If users are forced to bypass WfMS frequently, the system is more a liability than an asset [3]. This is why there have been many discussions on how to model and support processes that don't need to be predefined completely, but instead depend on and react to evolving circumstances and dynamic decisions by humans regarding a particular situation [3].

Also latest publications show the increasing interest of the BPM research community in unstructured, so-called "knowledge-driven" processes [4]. Such processes may differ from one execution to another showing huge unpredictability [3]. Knowledge-driven processes are also frequently referred to as "human-centric" processes [5]. Nevertheless, both terms imply the same meaning. It is meant that process execution significantly depends on knowledge of human experts rather than on completely predefined process models. In the following, we refer to such processes as "human-centric". Human-centric processes fundamentally differ from routine processes. These processes are significantly less structured. It is not possible to define the exact flow before a process is executed [2]. Supporting such processes with conventional WfMS has little prospect of success. Human-centric processes demand an agile, adaptable framework that is driven by creativity and empowers process performers to adapt to unpredictable circumstances rapidly [2].

Case Management (CM) and its extension Adaptive Case Management (ACM) are new trends in the field of BPM research that focus on the IT support of human-centric and therefore weakly structured processes. Unlike WfMS that use predefined process models to determine what should be done during a process, CM focuses on what *can* be done to achieve certain goals [3]. Thus, CM differs from the traditional view of structured and sequential predefined processes. CM assumes that workflows are nondeterministic, meaning they are driven by human decision-making during execution [1]. Applications of CM include licensing and permitting in government, patient care and medical diagnosis in healthcare, problem resolution in call centres, sales and operations planning, invoice discrepancy handling, maintenance and repair of machines and equipment, and engineering of made-to-order products [6].

Recently, the Object Management Group (OMG) released a first draft of the Case Management Modelling and Notation (CMMN) [6] that is intended to capture the common elements of CM research contributions. Referring to CMMN, human-centric processes are modelled by determine tasks that are applicable. Furthermore, CMMN allows to model rules that constrain the run-time execution of tasks. Therefore, process models in the context of CM consist of tasks and constraining rules and are therefore called *declarative* process models. Nevertheless, human-centric processes might also comprise sub-processes that are well structured. However, the overall process cannot be described by a predefined sequence of tasks. Therefore, managing human-centric processes often involves a mix of automatable routine work and human-centric work.

COLLABORATECOM 2013, October 20-23, Austin, United States
Copyright © 2013 ICST
DOI 10.4108/icst.collaboratecom.2013.254071

Although CM provides concepts that support human-centric work, processes have to be defined before they can be enacted by an IT system. Modelling processes is a cumbersome task [7]. Thus the first problem that arises when applying CM in practice is that especially in the beginning when organisations begin to formalize CM practice, it is often difficult to define rules which are an important part of declarative models controlling the applicability of tasks [8]. Sometimes it is even impossible to specify such rules beforehand and the company will have to "learn" them by doing in order to define them later. Therefore, declarative process models may first evolve based on the process participants' decisions [8].

Furthermore, researchers already discovered that fundamental complexity challenges arise when applying CM in practice [1]. By having the freedom to choose which task to perform next, the risk of confounding the users is taken. While not effectively using the provided freedom, users tend to feel overcharged instead [9].

In this contribution, we provide an integrated solution to the two problems described above, i.e., supporting the definition of processes as well as solving complexity challenges arising in the field of CM. Firstly, we propose that managing unstructured human-centric processes should start with CMMN model skeletons that serve as a lattice where initial process execution can leverage on. Secondly, by tracking different process instances based on participants' decisions, substantial process knowledge is recorded. Exploring process instance history might reveal certain recurring patterns that can be used in order to support users and therefore serve as dynamic guidance enhancement for CM systems. By exploring, i.e., mining recorded process history, process models might evolve over time which means that CMMN models are becoming more and more complete and ultimately reflect operational reality. For human-centric processes, declarative models are completed, i.e., constraining rules are abstracted, whereas for structured routine work traditional flow-oriented workflow models are abstracted.

At the end of this introduction, we will face again the most important questions focused by the work at hand:

What are the *research contributions* of this paper?

- By combining and adapting our recent technical innovations [10-11], our approach presents a practical solution to overcome complexity challenges arising when applying CM in practice.
- We foster that human-centric process management should start with CMMN model skeletons where initial process execution can lean against. By tracking and exploring process execution, process models might evolve over time. So, we alleviate the expensive and cumbersome process modelling phase.
- Besides, we provide a detailed mapping of conceptual and technical terms that occur in the field of latest BPM technology.

How does this *support collaborative work*?

- Knowledge-intensive work mainly consists of human interactions and therefore requires a high degree of collaboration. Humans need to be provided with tools to communicate, cooperate and coordinate information and activities [12].
- By simplifying the introduction of process management systems by process model evolvement and solving enactment issues by overcoming complexity challenges, our approach supports the integration of IT-based collaborative work in practice.

This paper is structured as follows: Section II gives a detailed overview of the different terms and concepts about less-structured human-centric processes that can be found in literature. In Section III, we shortly introduce CMMN as the new standard for modelling unstructured processes and describe how CMMN model skeletons serve as a lattice for initial process execution. Subsequently, Section IV explains our approach to learn different types of process models and patterns from process instance history. Section V describes the implementation of our approach. Section VI gives an overview of related work literature and the paper is finally concluded in Section VII.

II. ELABORATION OF TERMS AND CONCEPTS

ACM is a new trend in the field of BPM research [2] that focuses on the IT support of knowledge-intensive and therefore weakly structured, human-centric business processes. The central entity in ACM is the "case" which represents a human-centric process. Therefore, the case template and accordingly the case instance can be seen as the equivalent of a process model or process instance in BPM [14]. The general structure of ACM was derived from traditional CM, which has its origins in the juridical and medical field [13]. CM departs from the traditional view of structured and sequential predefined processes. ACM adapts these ideas and puts them into a broader context. ACM builds on the idea of goal-oriented collaboration and extends the traditional CM understanding by both overcoming CM's integration deficits with traditional WfMS and by introducing the concept of adaption. Thereby, process participants adapt to the individual requirements of a certain case. We adopt the ACM understanding, where the employee is empowered to perform these changes on his own [14]. In ACM, a process has two distinct phases, the design-time phase and the run-time phase. During the design-time phase, process analysts engage in modelling, which includes defining *mandatory* tasks that must always be part of a process and *optional* tasks that might be included into a process [6]. During the run-time phase, participants execute the model particularly by performing the mandatory tasks while the plan may continuously evolve due to the participants being engaged in planning when they add optional tasks to the plan of the process instance. Considering the example of Fig. 1, tasks A and B are always part of a process instance, whereas one or more instances of C and/or D can be added. "Planning at run-time" is a fundamental characteristic of CM [8].

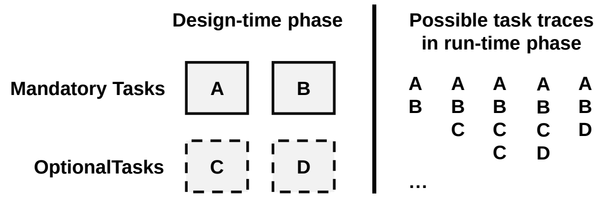

Figure 1. Overview of design-time and run-time phase of ACM based on [6]

After elaborating conceptual terms, we now face the technical realization of CM systems, i.e., process modelling paradigms and execution engines. Following the terminology of programming languages, there are two paradigms of describing business process models: the *imperative* and the *declarative* style. Imperative or procedural programming implies that every possible path must be foreseen at design time and encoded explicitly. If a path is missing then it is considered not allowed. Classic approaches like the BPEL [15] or BPMN [16] follow the imperative style and are therefore suited for predictable routine processes.

In declarative modelling, on the other hand, only undesired and forbidden paths and constellations are excluded so that all remaining paths are potentially viable. As the human-centric type of business processes often incorporates many unforeseen circumstances, the declarative approach is best suited [17]. Recently, the OMG released a first draft of the Case Management Modelling and Notation (CMMN) that intends to capture the common elements of CM and is declarative by nature [6].

Since managing human-centric processes often involves a mix of automatable routine and human-centric work, CM systems are technically based on declarative workflow models, e.g., CMMN, and execution engines as well as imperative workflow models, e.g., BPMN, and execution engines. Fig. 2 shows an overview of related concepts and technical terms in the field of weakly structured and human-centric process management.

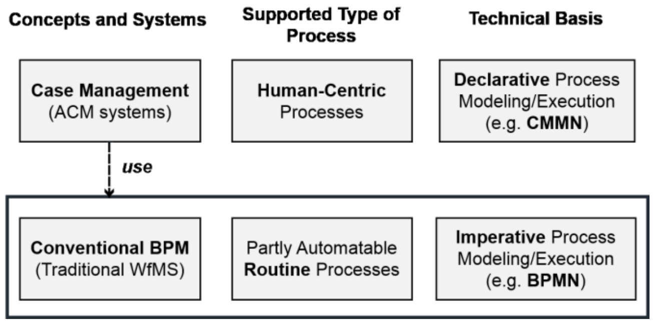
Concepts and Systems
Supported Type of Process
Technical Basis
Case Management (ACM systems)
Human-Centric Processes
Declarative Process Modeling/Execution (e.g. CMMN)
use
Conventional BPM (Traditional WfMS)
Partly Automatable Routine Processes
Imperative Process Modeling/Execution (e.g. BPMN)

Figure 2. Overview of related concepts and technical terms

III. MODELLING CMMN SKELETONS AND TRACKING EXECUTION

In this chapter, we will first introduce CMMN as an emerging standard for modelling human-centric business processes. Secondly, we will describe how CMMN skeletons can leverage initial process execution.

A. Case Management Modeling and Notation (CMMN)

A *Case* is a proceeding that involves actions taken regarding a subject in a particular situation to achieve a desired outcome [6], i.e., in our understanding a *Case* represents a human-centric process. The CMMN specification defines a meta-model, a notation and an XML serialization for case models [6]. CMMN modelling is typically concerned with determination of which tasks are applicable. Representation of the circumstances and the decision factors in a process model requires references to data. Furthermore, modelling of constraints on the tasks to be performed requires the specification of rules. In the following, we will give a short introduction to the most important elements of CMMN.

In CMMN, a *Case* is structured into *Stages* and *Tasks*. A Stage contains Tasks and may be considered an “episode” of a Case. A Case is the counterpart of a composite process or a subprocess in BPMN [16]. A Task may be a *HumanTask*, a *CaseTask* or a *ProcessTask*. A HumanTask is an atomic unit of work performed by a process participant, a CaseTask is used to reference another case and a ProcessTask is used to reference a business process modelled, e.g., in BPMN. In this way, it becomes possible to embed structured routine processes within human-centric processes. *CaseRoles* represent users or teams performing or adding Tasks collaboratively.

The information or references to information required to manage a Case is defined by a *CaseFile* and its *CaseFileItems*. The CaseFileItem may be considered the counterpart of the DataObject in BPMN [16]. Dependencies between Stages, Tasks and CaseFileItems are defined as *Sentries*. A Sentry defines a rule in the event-condition-action (ECA) structure [3]. It states that when a certain event of a Stage, Task or CaseFileItem is received and a certain condition on one or more CaseFileItems fulfils then a Stage or Task is entered or exited. The common structure of a Sentry is as follows:

```
ON «event» «Stage|Task|CaseFileItem»
IF «condition_CaseFileItem»
DO enable|exit «Stage|Task»
```

Tasks and Stages may be *discretionary* which means that they can be planned to the “discretion” of a process participant that is involved in planning. Discretionary items must be contained within the Cases *PlanningTable*.

The terms described above are the elements of CMMN we currently make use of in our approach. For a more detailed explanation of the CMMN standard, we refer to [6]. Fig. 3 shows an example CMMN model that comprises the different modelling elements. Here, the simplified process of writing a publication is defined. The model is depicted by a rectangle with its name in the upper left corner. The process definition consists of five different Tasks depicted by a rounded rectangle shapes. Three Tasks within the model are optional which means that only “Write Text” and “Publish Document” are mandatory to be performed. The shapes of the optional Tasks are drawn with dashed lines. The “Publish Document” Task is marked as a ProcessTask depicted by a chevron symbol in the upper left corner. Here, a structured routine sub-process, e.g., a BPMN process model is referenced by this Task. All other Tasks are HumanTasks depicted by a hand symbol in the upper left corner. Three Tasks are combined within the Stage “Prepare Document”. As already described, Tasks may have associated Sentries. When a Sentry is used as an entry criterion it is depicted by a shallow diamond shape. When a Sentry is used as an exit criterion it is depicted by a solid diamond shape. The diagram illustrates a situation where the entry of Task “Integrate Graphics” depends on the completion of Task “Create Graphics”. The underlying Sentry definitions (1) and (2) are as follows:

```
(1) ON complete “Create Graphics”
    DO enable “Integrate Graphics”
```

```
(2) ON complete "Create Graphics"
    DO enable "Generate List of Figures"
```

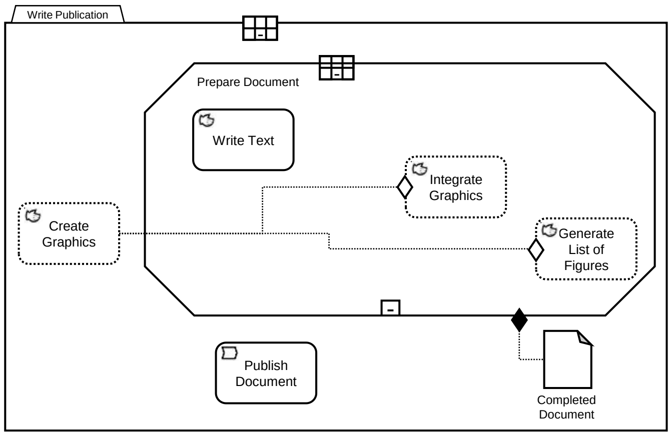

Figure 3. Example of a human-centric process modelled in CMMN [6]

B. *Modelling CMMN Skeletons*

The main part of CMMN modelling is concerned with the determination of tasks and the specification of rules, i.e., Sentries. We argue that it is often difficult to define all of these rules that control the applicability of tasks beforehand [8] and that it is even impossible to specify all rules beforehand and the company will have to "learn" them first in order to define them. As a result, we propose that managing unstructured human-centric processes should start with CMMN model skeletons that serve as a lattice where initial process execution can leverage on. A model skeleton contains Tasks, Stages and CaseFileItems that could have been defined beforehand. Fig. 4 shows a model skeleton that is related to the example in Fig. 3. Here, only four Tasks that are structured by a Stage and one CaseFileItem are defined beforehand.

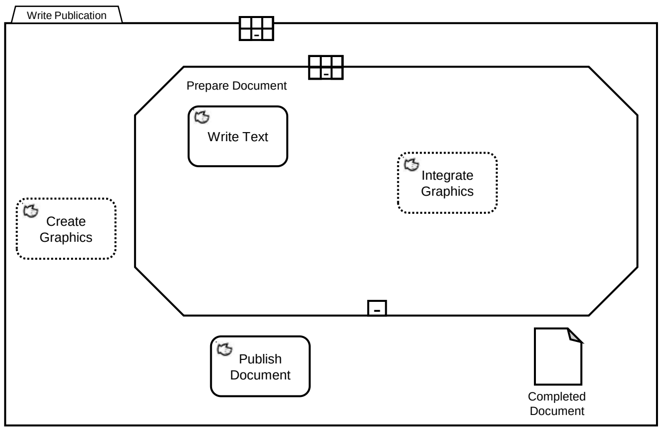

Figure 4. Model skeleton where initial execution can lean against

Note that no rules, i.e., Sentries used as entry or exit criteria, constraining the applicability of tasks have been defined yet. Additionally, the definition of the ProcessTask "Publish Document" has been omitted. We state that the information contained in this model skeleton can be provided with much less effort than modelling the complete process beforehand. Model skeletons can now serve as a first guideline for initial process execution.

C. *Tracking Process Execution*

Having modelled a CMMN skeleton, process participants can now engage in performing the work to be done with the help of an ACM system that enacts the defined model skeleton. This way, users can work on tasks and documents already offered by the system. The execution of mandatory tasks is ensured. Additionally, authorized users defined by CaseRoles can engage in adding missing tasks that are not yet part of the model. Note, that this functionality is very important in the field of human-centric work. Human experts should adapt to the individual requirements of a certain case and empowered to perform changes on their own. By using the ACM system while performing their work, process participants provide information about the process they are currently performing. Like traditional WfMS, ACM systems also allow for recording process execution to a process execution or event log [18]. By tracking different process instances based on participants' decisions, substantial process knowledge is recorded. By exploring, i.e., mining recorded process history, process models might evolve over time which means that CMMN models are becoming more and more complete and ultimately reflect operational reality. Table 1 shows a fragment of a process execution log that could have been assembled by recording process execution based on the model skeleton of Fig. 4. The log shows columns for the Task, Stage or CaseFileItem that the event relates to, the corresponding case instance, the type of event, i.e., whether a task has been started or completed, and the time of occurrence. Line 10, e.g., expresses that the "Completed Document" has been created. Line 11 denotes that the Stage "Prepare Document" has been completed.

TABLE I. A fragment of a process execution event log.

#	Task / Stage / CaseFile	Instance	Event Type	Time
1	**Prepare Document**	1 (WP)	Start	
2	Write Text	1 (WP)	Start	…
3	Write Text	1 (WP)	Complete	
4	Create Graphics	1 (WP)	Start	
5	Create Graphics	1 (WP)	Complete	
6	Integrate Graphics	1 (WP)	Start	
7	Integrate Graphics	1 (WP)	Complete	
8	Generate List of Figures	1 (WP)	Start	
9	Generate List of Figures	1 (WP)	Complete	
10	*Completed Document*	1 (WP)	*Created*	
11	**Prepare Document**	1 (WP)	Complete	
12	Publish Document	1 (WP)	Start	
	…			
19	Publish Document	1 (WP)	Complete	
20	**Prepare Document**	2 (WP)	Start	
21	Write Text	2 (WP)	Start	
22	Write Text	2 (WP)	Complete	
23	*Completed Document*	2 (WP)	*Created*	
24	**Prepare Document**	2 (WP)	Complete	
25	Publish Document	2 (WP)	Start	
26	Publish Document	2 (WP)	Complete	
…	**…**	…	…	

IV. Mining Case Instance History

By tracking different process instances based on participants' decisions, substantial process knowledge is recorded. In this chapter, we will describe how this information can be used to reduce complexity of ACM systems as well as to support modelling of processes. We distinguish three different types of mining methods process participants and analysts can leverage: mining *best-practice patterns*, mining human-centric processes, i.e., *declarative process models* and mining structured routine processes, i.e., *imperative process models*.

Firstly, exploring process instance history might reveal recurring patterns that can be used to support users and therefore serve as dynamic guidance that reduces complexity in ACM systems. Therefore, the approach at hand makes use of the traditional data mining technique of association rule mining.

Secondly, by the use of process mining techniques it becomes possible to discover process models - declarative as well as imperative - automatically [18]. The idea of process mining is to extract knowledge from event logs recorded by WfMS. The computer-aided creation of process models offers huge potential of saving time [19]. By deriving process models from event logs, the appropriateness of process models can be guaranteed to a certain extent, since they are constructed according to the way the processes have actually been executed [19].

A. Discovering Recurring Best-Practice Patterns

We present a practical solution to overcome complexity challenges arising when applying CM in practice. We assume that the participating human experts usually know best which specific process setting to apply in a given situation [2]. Additionally, we assume that performing agents tend to improve their process execution to achieve higher efficiency, quality or reputation. By relying on the employees' innovative spirit we can build something we call a self-optimizing system. If an employee finds an alternative task sequence that fits better into certain situations, other agents can collaboratively benefit from this experience by following the generated recommendations [10].

Therefore, our approach discovers best-practice patterns that provide guidelines through the human-centric process. As the data basis, i.e., the event log grows with progressing enactment of ACM systems, quantity and quality of discovered patterns will dynamically change as well. We adapt an association rule mining algorithm to analyse process execution logs. The general idea is part of our previous work [11]. However, we refine the approach leading to a generic method suitable for different fields of application. The resulting association rules, i.e., best-practice patterns are used for operational support and thus for guiding process participants through process execution. Therefore, process event logs are periodically transformed to an input dataset for association rule mining.

The transformation procedure depends on the size w of the sliding window. The longer the sliding window, the further the foresight. Let $w = 2$, the algorithm will extract patterns that look exactly one task ahead. Fig. 5 visualizes the main steps of the transformation procedure. The figure shows an example of three different process instances. Within the first two instances five tasks have been executed: B after A, C after B and so on. The third instance contains only two tasks with E succeeding A. Assuming a sliding window with $w = 2$, four itemsets have been extracted from the instances 1 and 2 respectively. From instance 3 only one itemset has been generated. The numbers in brackets represent the corresponding time steps within the sliding window respectively. Summing up, the generated dataset that serves as input for association rule mining contains 9 itemsets in this example.

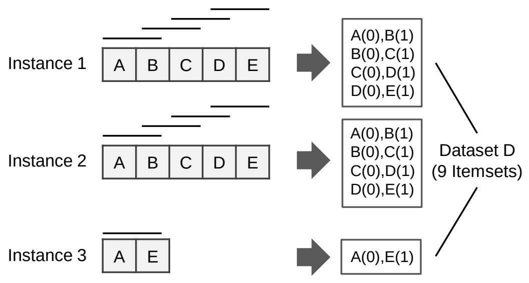

Figure 5. Transformation of event logs to dataset

Referring to the example in Section III, the tasks A to E could encode the tasks "Write Text", "Create Graphics" and so on. Subsequently, the well-known data mining algorithm *Apriori* [20-21] is applied to the dataset D. The algorithm extracts a set of association rules. Rule (3), e.g., claims that when task "Create Graphics" has been performed, task "Integrate Graphics" followed frequently.

```
(3) "Create Graphics"(0) ⟶ "Integrate
    Graphics"(1)
```

If a user's behaviour satisfies the left-hand side of a rule the right-hand side of the rule is recommended. Process participants using the ACM system still have the freedom to choose which task to perform next. However, best-practice patterns serve as guidelines that help users to negotiate their way and thus reduce complexity.

Since human-centric processes especially involve data usage and data flows [1], we additionally extended our approach to comprise the recording of consumed and produced data items. Therefore, a single record of an event log additionally contains columns for data input and data output, i.e., consumed and produced documents or data fields. The transformation procedure is implemented analogous. Fig. 6 for example shows how the event records of process instance 3 are enriched with produced and consumed data. Within the itemset, the additional data-related information is simply appended. The numbers in brackets again represent the corresponding time steps.

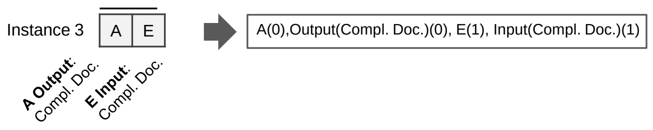

Figure 6. Transformation of event log to data-enriched itemset

Based on the described data-enriched dataset, extracted association rules, e.g., the usage of documents are recommended. Rule (4), e.g., claims that the task "Publish Document" frequently "consumed", i.e., required, the data item "Completed Document".

```
(4) "Publish Document"(1) ⟶ Input("Completed
    Document")(1)
```

B. *Mining Less Structured Human-Centric Work*

Until now, process execution has been supported by best-practice patterns and was based on a CMMN model skeleton. Tracking process execution for a sufficient time period yields an extensive event log containing substantial process knowledge. The size of the corresponding time period depends on the application's dimension. When sufficient process information has been recorded, process analysts initiate a "snapshot" regarding the accumulated information. This snapshot is used to complete existing process model skeletons by mining process models from the recorded information.

First of all, we focus on less-structured processes. As already mentioned, the declarative process modelling approach is best suited for modelling less-structured human-centric work. CMMN is an example of a declarative process modelling language. As described in Section III, the building blocks of CMMN are entities, i.e., Tasks and CaseFileItems, as well as rules, i.e., Sentries. Thus, mining or "reconstructing" CMMN models from event logs comprises two issues: defining entities and observing rules with respect to the underlying event log.

1) *Defining Tasks and CaseFileItems*

The first issue, i.e., defining entities, can be faced in a simple way. Entities have already been defined beforehand in the CMMN model skeleton or have been added by process participants during run-time through adaption. The newly added entities will be taken over into the CMMN model skeleton. Considering the model skeleton of Fig. 4 and the example event log of Table I, the task "Generate List of Figures" (lines 8 and 9) has obviously been added at run-time. Therefore, it will be adopted into the resulting CMMN model as a HumanTask. If a Task is performed in every recorded process instance, it will be defined as mandatory, otherwise it will be defined as optional.

2) *Mining Sentries*

Mining rule-based process models consists in observing the compliance of rules [22]. The approach at hand is suitable for mining any rule-based, i.e., declarative process modelling language. For CMMN it is necessary to translate Sentries to constraint templates that can be observed with respect to the provided event log. As stated above, a Sentry is an (event, condition, action) triple defining that on a certain event (of a Task, Stage or CaseFileItem) and if a certain condition (on a CaseFileItem; optional) evaluates to true a certain Task or Stage is enabled or exited. In our declarative mining approach constraint-templates are formulated as logical statements. Thus, the Sentry

```
ON «event» «Stage|Task|CaseFileItem»
IF «condition_CaseFileItem»
DO enable|exit «Stage|Task»
```

is translated to the constraint template

```
enable|exit(«Stage»|«Task») ⟶
«event»(«Stage|Task|CaseFileItem») ∧
«condition_CaseFileItem»
```

The constraint template corresponding, e.g., to rule (1) is the logical statement (5).

```
(5) enable(«Task») ⟶ complete(«Task»)
```

A constraint-template can also contain events that relate to CaseFileItems. Constraint (6), e.g., expresses the logical statement of a Sentry as an exit criterion where the completion of a Stage implies that a certain CaseFileItem has been created.

```
(6) exit(«Stage») ⟶ created(«CaseFileItem»)
```

The first step is to define *all possible* constraints that can be assembled based on the provided constraint templates as well as the provided entities. Consider, e.g., constraint template (5) where the template consists of two events that relate to a Task respectively. Thus, a constraint for every possible combination of Tasks that have been defined, is generated. Based on the example event log of Table I, five Tasks have been defined. Therefore 25 different constraints will be generated initially. Based on these constraints, we observe the provided event log. Here, we make use of the principal of proof by contradiction. Hence, every rule is negated. If the negation can be proven within the event log, the corresponding constraint is neglected. Finally, the procedure returns constrains that have not been neglected during the observation of the provided event log.

The functionality of the method is shown by an example. Rule (7) and (8) express *two exemplary* constraints that have been generated in the first step of the procedure based on template (5).

```
(7) enable("Integrate Graphics") ⟶
    complete("Create Graphics")

(8) enable("Write Text") ⟶ complete("Create
    Graphics")
```

Before searching the event log, both constraints are negated. Therefore, rule (9) expresses the negation of rule (8).

```
(9) enable("Write Text") ∧ ¬complete("Create
    Graphics")
```

Colloquially spoken, the method is searching for a counterexample, i.e., a process instance where Task "Write Text" was started while Task "Create Graphics" has not been completed yet. Considering the event log in Table I, instance 2 contains such a case since "Write Text" has been started without the completion of "Create Graphics". Thus, constraint (8) will be neglected whereas for constraint (7) no counterexample can be found.

Finally, a Sentry definition will be generated for every remaining constraint, i.e., the remaining constraints are translated to Sentries. Thus, considering the model skeleton of Fig. 4, the example event log in Table I and the constraint templates (5) and (6), the method described above would complete the model skeleton by discovering the three Sentries as shown in Fig. 3. Note, that our approach again also considers data objects and is not restricted to task sequences. By providing data-centric templates as shown in (6), our approach discovers Sentries that relate to CaseFileItems as well. Sentry (10), e.g., defines an exit criterion for the Stage "Prepare Document" that depends on the creation of a document.

```
(10)  exit("Prepare Document") ⟶
    created("Completed Document")
```

C. Abstracting Structured Routine Work

In cases where parts of the recorded information represent a structured sub-process, i.e., a *ProcessTask*, where every possible path can be described clearly, the automatic generation of an imperative process model expressed, e.g., in BPMN can be initiated. There are several well-known imperative process mining techniques that can be used to generate this kind of model [7,23]. In this chapter, we will show how we can leverage these methods to specify ProcessTasks, i.e., well-structured sub-processes within a human-centric process described by a CMMN model.

The model skeleton of Fig. 4 defined the task "Publish Document" as an atomic HumanTask. However, while performing the process participants might have engaged in further planning and added/adapted sub-tasks within the task "Publish Document". Consider Table II that shows an event log fragment that has been recorded when performing the example process of Section III. Here, obviously three sub-tasks have been added to the task "Publish Document" (PD) until it has been completed. Since users added sub-tasks to "Publish Document" it does not represent an atomic HumanTask anymore but contains a sub-process that can be further defined by a process model of its own.

TABLE II. Example event log reflecting a structured sub-process.

#	Task / Stage / CaseFileItem	Instance	Event Type	Time
…	**…**	…	…	…
12	**Publish Document**	**1 (WP)**	Start	
13	Generate PDF Document	1 (PD)	Start	
14	Generate PDF Document	1 (PD)	Complete	
15	Check Visualization	1 (PD)	Start	
16	Check Visualization	1 (PD)	Complete	
17	Submit Document	1 (PD)	Start	
18	Submit Document	1 (PD)	Complete	
19	**Publish Document**	**1 (WP)**	Complete	
…	**…**	…	…	…

The process of publishing a document is an example of structured routine work, since this work will be performed similarly in all instances. Although, the example log fragment in Table II only shows the event sequence of one instance of the sub-process "Publish Document", it is likely that further recorded instances would have the same event sequence. Taking the recorded event log of such structured routine work as a basis, the formerly atomic HumanTask "Publish Document" can be redefined as a ProcessTask that invokes a traditional flow-oriented workflow when enabled. Using one of the already mentioned mining methods, we can extract an imperative process model, e.g., in the BPMN notation. Fig. 7 depicts the model that could has been extracted in consideration of the event log in Table II. Furthermore, the CMMN model in Fig. 4 will change as well as, since the ProcessTask is now depicted by a chevron symbol in the upper left corner.

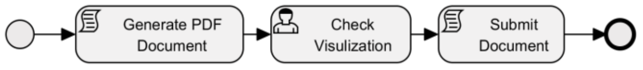

Figure 7. Extracted structured process as a BPMN model

Summing up, what did we achieve? First of all, a human-centric process does not have to be modelled completely beforehand. Instead, process analysts start by modelling CMMN skeletons that contain well-known information like mandatory tasks. This kind of information can be provided with much less effort than modelling every possible specification. Subsequently, process participants engage in performing their work accompanied by tracking the execution and the corresponding events. In this way, substantial process knowledge is composed. This knowledge is used in three different ways:

- We extract best-practice patterns that serve as guidelines for future process execution. Users can leverage these patterns without modelling any process beforehand. Note, that these patterns could also be interesting even though the process has been modelled completely.
- The modelling phase of human-centric processes, i.e., a declarative process models is supported by extracting applicability rules, i.e., Sentries of CMMN models.
- The modelling phase of partly structured routine sub-processes, i.e., imperative process models is supported by generating traditional flow-oriented workflow model, e.g., in BPMN.

V. Implementation

In this chapter, we will describe how we implemented our approach. Therefore, we present two systems, the *Process Observation* system and the *Process Navigation* system.

A. Tracking Execution and Learning Process Models by Process Observation

The application of our system starts by defining a CMMN model skeleton where tasks that could be defined beforehand are divided into mandatory and optional tasks. These tasks are assigned to processes that can contain sub-processes in turn. Based on this task definition, the users can now engage in performing the process accompanied by tracking the execution into an event log. Fig. 8 shows the event log's meta-model used by Process Observation. The class diagram shows the main parts of the data model. The central entity of the data model is the Event that is a generalization of CaseFileEvents and TaskEvents. These two classes are linked to transitions that are defined in the CMMN lifecycle of CaseFileItems as well as PlanItems, e.g., Tasks. Further entities are the process performers (usually human participants) that execute Tasks or create/alter CaseFileItems. Furthermore, CaseFileItems with a certain value may be produced or consumed. When a process is executed an instance is created. Tasks may be started and completed by a performer.

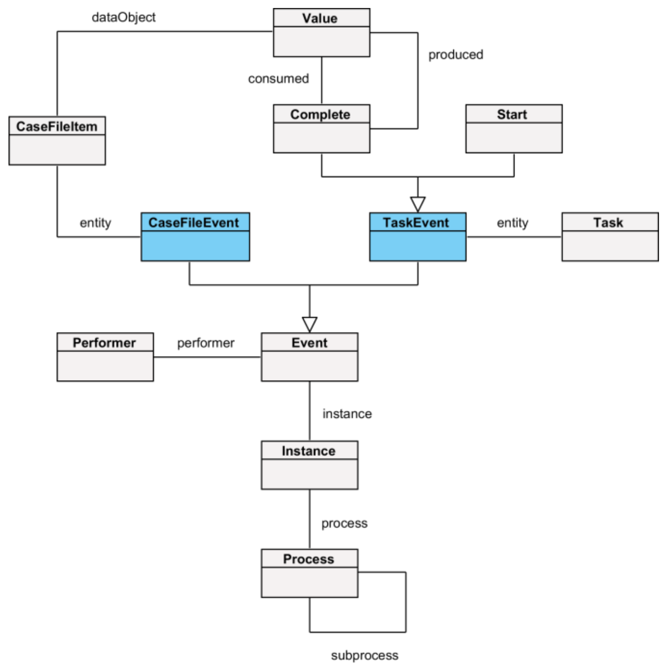

Figure 8. Meta-model of the process execution log

Using the recorded process information, the Process Observation system extracts best-practice patterns as well as imperative and declarative process models. In order to extract best-practice patterns the event log is transformed to a dataset as already described in Section IV. Here, only the *PlanItemTransition* records are considered for transformation. The transformed itemsets form the input dataset for the well-known Apriori algorithm [20-21]. After applying the Apriori algorithm the system manages the currently extracted association rules. If a user's behaviour satisfies the left-hand side of a rule the right-hand side of the rule is recommended. Fig. 9 e.g., depicts the recommendation to continue the process by performing the task "Publish Document". Note that the recommended action is only a guideline and is not required to be followed.

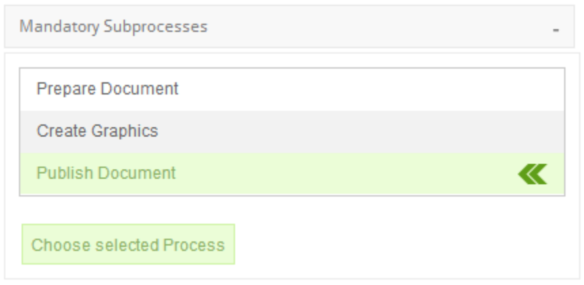

Figure 9. Screenshot showing a recommended action

For the implementation of imperative process mining algorithms we refer to the well-known imperative mining techniques [7,23]. We adapted the existing algorithms that generate process models as Petri nets to extract more practical process models in BPMN notation.

The declarative process mining approach of the work at hand is implemented by the *Drools* platform [24]. Therefore, the logical statements that represent Sentries are transformed into the *Drools Rule Language* (DRL) rules that can be executed by the Drools solver. The Drools solver observes the event log and extracts constraints by proving the occurrence of negated rules. The defined artefacts, i.e., Tasks and CaseFileItems, combined with the extracted Sentries form a CMMN process model. The generated process models are stored within a common repository.

B. *Enactment of CMMN and BPMN Models by Process Navigation*

As stated above, the full business process range includes both the structured routine and the human-centric processes. The goal of the *Process Navigation* system (PN) is to provide a platform for execution both types of processes. The system is designed as a virtual machine for executing both imperative and declarative process models. The imperative part is based on the Process Virtual Machine (PVM) pattern [25] which is applied, e.g., in the Activiti engine. The declarative part is based on an event-driven execution engine described in [5].

The declarative part natively interprets the so-called *Declarative Process Base Language* (DPBL) which the CMMN model is mapped to. Fig. 10 shows the relevant structural entities in DPBL.

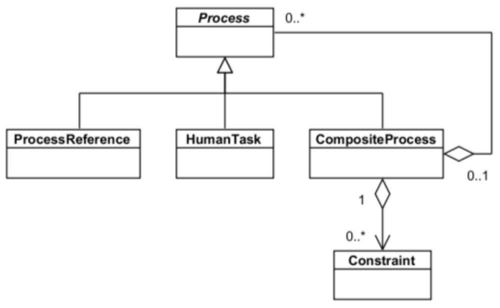

Figure 10. Excerpt of the DPBL meta-model

For the structural part it is necessary to map the CMMN entities to DPBL entities. Stages and CaseTasks are mapped to composite *CompositeProcesses*, a HumanTask remains a *HumanTask*, a ProcessTask is a *ProcessReference* and a Sentry maps to a *Constraint*.

For the rule part it is necessary to translate Sentries to *Constraints*. As stated above, a Sentry is an (event, condition, action) triple defining that on a certain event (of a Task, Stage or CaseFileItem) and if a certain condition (on a CaseFileItem; optional) evaluates to true then a certain Task or Stage is entered or exited. In DPBL constraints are formulated as first-order logic statements analogous to the definition of constraint templates described in Section IV. These constraints on the process instance's event trace must hold during the instance's execution.

Summing up, Fig. 10 illustrates the application spectrum of the described approach and systems of the work at hand. The Process Observation module comprises the enactment of model skeletons, the tracking of initial process execution, the extraction of best-practice patterns and finally the generation of process models through process mining methods. The Process Navigation module enacts the completed process models, i.e., it provides an engine for declarative as well as for imperative process models. Therefore, it embraces WfMS functionality for both types of process models. Combining both modules, the comprehensive implementation represents an elaborate ACM system that additionally contains guidance functionality as well as initial process modelling support.

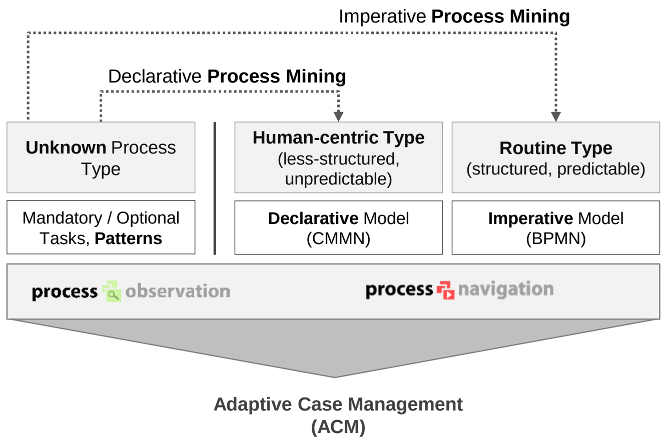
Imperative Process Mining
Declarative Process Mining
Unknown Process Type
Human-centric Type (less-structured, unpredictable)
Routine Type (structured, predictable)
Mandatory / Optional Tasks, Patterns
Declarative Model (CMMN)
Imperative Model (BPMN)
process observation
process navigation
Adaptive Case Management (ACM)

Figure 11. Overview of process types, model paradigms and systems

VI. Related Work

Adaptive Case Management (ACM) reflects a more flexible, dynamic approach to supporting work [2]. Instead of predefining every possible process step or path, ACM systems allow process participants to dynamically instantiate processes from templates as well as newly occurring processes when needed. The main concepts and research achievements in the context of ACM can be found in the publication “Mastering the Unpredictable” [2]. The publication “Taming the Unpredictable” [26] additionally includes a collection of case studies that present real-life implementations of the ACM approach. There are already mature implementations of ACM, e.g., [9,14]. However, existing ACM approaches lack the use of recorded information and the integration with WfMS [27]. In [1] the authors give an elaborate compendium of common ACM principles and introduce the most important complexity challenges arising in the field of ACM. The specified solution approaches mainly focus on improving visualization, personalization and customization of ACM dashboards. Instead of improving visualization, we face the complexity problem by comprising recorded process history. In cases where users need some guidelines in the flexible, unstructured environment, workflow pattern mining methods can be used to find previously unknown coherencies. Therefore, well-known representatives are sequence [28] or episode mining [29] that extract frequently occurring fragments of processes. However, these methods are limited to the extraction of coherencies considering the execution order of process steps. Other types of process information, like incorporated data or agents are again neglected.

Using the recorded process execution data, latest pattern recognition methods offer the possibility to extract complete process models [18]. Van der Aalst et al. developed techniques and applied them in the context of workflow management under the term Process Mining [18]. There are already several algorithms [7,23] and even complete tools, like the ProM Framework [30], that aim at generating process models automatically. During the last decade, several algorithms have been developed, focusing different perspectives of process execution data.

Many of these traditional process mining algorithms are imperative approaches. These methods construct imperative models explicitly showing all possible behaviours. Other ways to mine for process models are declarative approaches. Instead of explicitly specifying all the allowed sequences of events, declarative process models specify the possible ordering of events implicitly by constraints. There are several declarative process mining algorithms like [22,31,32], however, these methods only focus the sequence of process steps. Incorporated data or agents are neglected. Especially in fields of human-centric processes that are frequently driven by data usage, declarative process models should comprise more than just control-flow specific rules.

Extracted process models can finally be enacted by workflow engines. In addition to traditional flow-oriented, i.e., imperative, engines [25], declarative workflow engines enact provided process models. The most recent approach in the field of declarative process management is the Declare framework [33]. It is based on linear temporal logic (LTL) and therefore allows for relating process steps by temporal and existential constraints. These constraints may not contain statements on data, agents or tools. The only way of relating the temporal order of steps to these perspectives is to make the constraints depend on certain conditions. Such a conditional constraint only applies if its condition evaluates to true. Though a condition could then contain statements on data, agents and tools, the actual constraint remains limited to temporal order and existence of steps. The other perspectives cannot be constrained, which reduces the expressivity of the supported process modelling languages.

VII. Conclusion, Limitations and Outlook

In this contribution, we introduced an integrated solution to two issues arising in the field of supporting human-centric processes, i.e., supporting the definition of processes as well as solving complexity challenges. We proposed that managing unstructured human-centric processes should start with CMMN model skeletons that serve as a lattice where initial process execution should leverage on. By tracking different process instances based on participants’ decisions, substantial process knowledge is recorded. Subsequently, event logs can be analysed and might reveal certain recurring patterns that can be fed back to users and therefore serve as dynamic guidance enhancement for ACM systems. Furthermore, by mining recorded process history, process models might evolve over time, i.e., CMMN models are becoming more and more complete and reflect operational reality. For human-centric processes, declarative models might be abstracted, i.e., constraining rules have been abstracted, whereas for structured routine work traditional flow-oriented workflow models can been abstracted.

A drawback of our approach is that currently process analysts have to know whether a process is structured routine work or less-structured knowledge-driven work. Based on this knowledge the corresponding mining method must be chosen, i.e., mining a declarative or an imperative process model.

Therefore a further objective is to develop heuristics that might support analysts in choosing the right mining method. Moreover, discovering best-practice patterns using the Apriori algorithm may result in a high number of association rules. Many of them are trivial like, e.g., the fact that the performer of a task is always a person. A future task is to reduce the number of extracted rules by identifying the trivial one.

ACKNOWLEDGMENT

The presented prototypes are developed and used in the project "Kompetenzzentrum für praktisches Prozess- und Qualitätsmanagement", which is funded by "Europäischer Fonds für regionale Entwicklung (EFRE)".

REFERENCES

[1] Huber, S., Hauptmann, A., Lederer, M., Kurz, M. (2013). Managing Complexity in Adaptive Case Management. In S-BPM ONE-Running Processes (pp. 209-226). Springer Berlin Heidelberg.

[2] Swenson, K. D. (2010). The Nature of Knowledge Work. In: Swenson, K.D. (ed.) Mastering the Unpredictable. 63-88. Meghan-Kiffer Press, Tampa.

[3] Van der Aalst, W. M., Weske, M., & Grünbauer, D. (2005). Case handling: a new paradigm for business process support. Data & Knowledge Engineering, 53(2), 129-162.

[4] Reijers, H. A., Slaats, T., Stahl, C. (2013). Declarative Modeling–An Academic Dream or the Future for BPM?. In Business Process Management (pp. 307-322). Springer Berlin Heidelberg.

[5] Zeising, M., Schönig, S., Jablonski, S. (2012). Improving collaborative business process execution by traceability and expressiveness. In Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2012 8th International Conference on (pp. 435-442). IEEE.

[6] Object Management Group (OMG), Case Management Model and Notation (CMMN), Version 1.0 – Beta 1 (2013). Last Access: July 2013. Available: http://www.omg.org/spec/CMMN/1.0/Beta1.

[7] Van der Aalst, W., Weijters, T., Maruster, L. (2004). Workflow mining: Discovering process models from event logs. Knowledge and Data Engineering, IEEE Transactions on, 16(9), 1128-1142.

[8] De Man, H. (2009). Case management: Cordys approach. BPTrends, February, 1-13.

[9] Kurz, M., Herrmann, C. Adaptive Case Management-Anwendung des Business Process Management 2.0-Konzepts auf schwach strukturierte Geschäftsprozesse. Dienstorientierte IT-Systeme für hochflexible Geschäftsprozesse, University of bamberg Press, Bamberg. 241-265.

[10] Günther, C., Schönig, S., Jablonski, S. (2012). Dynamic guidance enhancement in workflow management systems. In Proceedings of the 27th Annual ACM Symposium on Applied Computing (pp. 1717-1719). ACM.

[11] Schönig, S., Zeising, M., Jablonski, S. (2012). Adapting association rule mining to discover patterns of collaboration in process logs. In Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2012 8th International Conference on (pp. 531-534). IEEE.

[12] Ukelson, J.P (2010). What to do when modeling doesn't work. In: Swenson, K.D. (ed.) Mastering the Unpredictable. pp. 29–39. Meghan-Kiffer Press, Tampa.

[13] Matthias, J.T. (2010). Technology for Case Management. In: Swenson, K.D. (ed.) Mastering the Unpredictable. pp. 63–88. Meghan-Kiffer Press, Tampa.

[14] Herrmann, C., Kurz, M. (2011). Adaptive Case Management: Supporting Knowledge Intensive Processes with IT Systems. In S-BPM ONE-Learning by Doing-Doing by Learning (pp. 80-97). Springer Berlin Heidelberg.

[15] Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I., Weerawarana, S. (2003). Business Process Execution Language for Web Services - Version 1.1.

[16] Object Management Group Inc. (2011). Business Process Model and Notation (BPMN) Version 2.0, http://www.omg.org/spec/BPMN/2.0.

[17] Fahland, D., Lübke, D., Mendling, J., Reijers, H., Weber, B., Weidlich, M., Zugal, S. (2009). Declarative versus Imperative Process Modeling Languages: The Issue of Understandability. 10th International Workshop BPMDS 2009, and 14th International Conference, EMMSAD 2009, held at CAiSE 2009). pp. 353–366. Springer, Amsterdam, Netherlands.

[18] Van der Aalst, W. (2011). Process Mining: Discovery, Conformance, and Enhancement of Business Processes, Springer, Berlin-Heidelberg, DOI 10.1007/978-3-642-19345-3, ISBN: 978-3-642-19344-6.

[19] Schönig, S., Günther, C., Jablonski, S. (2012). Process Discovery and Guidance Applications of Manually Generated Logs. In ICIMP 2012, Seventh International Conference on Internet Monitoring and Protection (pp. 61-67).

[20] Agrawal, R., Imieliński, T., Swami, A. (1993). Mining association rules between sets of items in large databases. In: Proceedings of the 1993 ACM SIGMOD international conference on Management of data, ACM, New York, USA. ISBN: 0-89791-592-5, DOI: 10.1145/170035.170072.

[21] Zhang, C., Zhang, S. (2002). Association rule mining: models and algorithms. Springer, Berlin-Heidelberg, ISBN: 3-540-43533-6.

[22] Maggi, F. M., Mooij, A. J., van der Aalst, W. M. (2011). User-guided discovery of declarative process models. In Computational Intelligence and Data Mining (CIDM), IEEE. (pp. 192-199).

[23] Weijters, A., Van der Aalst, W., and De Medeiros, A. (2006). Process mining with the heuristics miner-algorithm. Department of Technology, Eindhoven University of Technology, Eindhoven, Netherlands.

[24] JBoss. (2013, 07/30/2013). Drools Expert User Guide Version 5.4.0.Final. Available:http://docs.jboss.org/drools/release/5.4.0.Final/drools-expert-docs/html/

[25] T. Baeyens and M. V. Faura. (2007). The Process Virtual Machine. Available: http://docs.jboss.com/jbpm/pvm/article/

[26] Swenson, K.D. (2011). Taming the Unpredictable. Real World Adaptive Case Management: Case Studies and Practical Guidance. Future Strategies Inc., Lighthouse Point.

[27] Motahari-Nezhad, H. R., Bartolini, C., Graupner, S., Spence, S. (2012). Adaptive case management in the social enterprise. In Service-Oriented Computing (pp. 550-557). Springer Berlin Heidelberg.

[28] Srikant, R., Agrawal, E. (1996). Mining Sequential Patterns: Generalization and Performance Improvements. 5th International Conference on Extending Database Technology (EDBT '96). pp. 3–17.

[29] Mannila, H., Toivonen, H., Verkamo, A.I. (1997). Discovery of Frequent Episodes in Event Sequences. Data Mining and Knowledge Discovery. 1, 259–289.

[30] Van Dongen, D., De Medeiros, A., Verbeek, H., Weijters, A., Van der Aalst, W. (2005). The ProM framework: A new era in process mining tool support. In: Ciardo, G., Darondeau, P. (eds.) Applications and Theory of Petri Nets, LNCS, vol. 3536, pp. 444-454.

[31] Chesani, F., Lamma, E., Mello, P., Montalo, M., Riguzzi, F., Storari, S. (2009). Exploiting Inductive Logic Programming Techniques for Declarative Process Mining. Transactions on Petri Nets and Other Models of Concurrency II. 5460, 278–295.

[32] Bellodi, E., Riguzzi, F., Lamma, E. (2010). Probabilistic Declarative Process Mining. Knowledge Science, Engineering and Management. 6291, 292–303.

[33] Pešić, M. (2006). Constraint-Based Workflow Management Systems: Shifting Control to Users.