Notation-agnostic Subprocess Modeling for Adaptive Case Management

Johannes Tenschert
Friedrich-Alexander-Universität Erlangen-Nürnberg

Sebastian Dunzer
Friedrich-Alexander-Universität Erlangen-Nürnberg

Martin Matzner
Friedrich-Alexander-Universität Erlangen-Nürnberg

Notation-agnostic Subprocess Modeling for Adaptive Case Management

Johannes Tenschert, Sebastian Dunzer, and Martin Matzner

Chair of Digital Industrial Service Systems,
University of Erlangen-Nürnberg, Nürnberg, Germany
{johannes.tenschert,sebastian.dunzer,martin.matzner}@fau.de

Abstract. Even though knowledge work comprises tasks that cannot be modeled a priori, some structure for routine work or an outlined course of action is often necessary. Knowledge workers perform structured and ad-hoc activities – and a wide range of requirements typically yields many tools. To avoid redundant and scattered process information, a single system of record capable of consolidating flexibility and modeling of different aspects of processes is desirable. For knowledge-intensive processes, we outline how to combine process models for structured routine aspects with the ad-hoc activities and flexibility of social software in the same context. Therefore, a case comprises modeled subprocesses as well as ad-hoc activities and artifacts. The shared context allows to transparently combine aspects and deviate from predefined models. We prototypically implement our approach and show that ad-hoc project management and different notations can be applied within the same case.

Keywords: adaptive case management, subprocess modeling, knowledge-intensive business process, process flexibility

1 Introduction

In recent years, the share of knowledge work in the workforce rapidly increased, and knowledge-intensive processes became customary. In the US, around 50% of the work can be attributed to knowledge work, and other countries show similar tendencies [1], [2]. Knowledge workers are responsible for their own contribution in terms of quantity and quality [3], and they perform emergent processes [4]. Flexibility requirements often yield multiple support systems for the same process, e. g. groupware, collaboration tools, and business process management (BPM) systems. These need to be integrated or manually kept consistent. Otherwise there is no clear system of record. Adaptive case management (ACM) systems aid within this realm.

Business process modeling typically entails capturing the whole process. This approach increases efficiency for predictable and frequent processes. For knowledge-intensive processes with a lot size of 1, and for unstructured work, this sort of process support is detrimental. Modeling scarcely executed processes often cannot be amortized with efficiency gains. Moreover, different processes of an organization might be appropriate for different modeling notations or paradigms. Current techniques only allow mixing paradigms to some degree.

This paper introduces an approach to combine structured process models for routine aspects of knowledge-intensive processes with ad-hoc activities and artifacts within the same context. Our approach is notation-agnostic and thereby permits mixing modeling languages within the same overall case. Knowledge workers can transparently deviate from predefined models or decide to not apply them at all, e. g. by creating and adapting tasks and other artifacts that have not been modeled in advance. The approach facilitates creating a single system of record. We apply cases as the context for all routine and ad-hoc work. Process models are applied as subprocess models using data from this context. Subprocess models can be combined as they are initiated on demand. All related case artifacts are stored or referenced within the same case. We prototypically implement our approach and a motivating example in the commercial ACMS Pertuniti.

The following sections introduce fundamentals in regard to ACM and process modeling paradigms, and a motivating example in the context of providing a lecture. Section 6 captures our approach on notation-agnostic subprocess modeling, and Section 7 outlines the implementation in the commercial ACMS Pertuniti. Afterwards, we discuss the approach and introduce related work. Finally, we conclude and outline further work.

2 Fundamentals

Since we combine structured process models with flexibility of ACM, our approach has to consider many existing techniques. Hence, we first introduce ACM and outline modeling paradigms we consider as applicable for subprocess models.

2.1 Adaptive Case Management

The term *adaptive case management* (ACM) has been introduced in *Mastering the Unpredictable* [5]. ACM systems support knowledge workers that perform emergent or unstructured work. The actions performed in these processes are typically not known in advance [4]. The focal point for organizing the work is the *case*. Typical artifacts are ad-hoc tasks and unstructured notes or documents. As design and execution are the same phase, an ACM system has to support knowledge workers in planning their emergent processes.

In contrast to process-driven applications, support systems in case management must consider flexibility and ad-hoc activities, the focus on documents and unstructured data, and empowering knowledge workers to deviate from predefined models to support new situations in emergent processes. Still, knowledge work also comprises some routine work that our approach intends to reduce.

2.2 Process Modeling Paradigms

Today, business process models can be created in a wide range of modeling paradigms that entail different benefits and limitations.

Declarative. Declarative process modeling became prevalent for ACM [6], [7]. It is an approach to create flexibility in processes to support less predictable courses of action [7], [8], [9]. Although knowledge work is inherently unpredictable, some aspects *can* contain predictable dependencies. Declarative notations focus on dependencies [10], [11]. Declarative models allow the execution of any modeled activity, unless a constraint prevents it [6]. They explicitly prohibit behavior and implicitly describe applicable paths. There is a wide range of declarative process modeling notations, e. g. Declare [12], CMMN [13], DCR graphs [14], and DPIL [15].

Imperative. In contrast, imperative approaches describe all permitted paths [16]. They implicitly prohibit behavior [17]. Imperative models focus on describing sequences of actions [8]. Subsequently, they can be considered detrimental for ACM [18]. Depending on the level of detail, such models enable process automation [16]. Even though process mining leveraged the generation of process models from data, imperative modeling is time- and resource-intensive [19]. Routine tasks that knowledge workers perform can be improved or even automated with imperative approaches. Prominent examples of imperative process modeling languages comprise BPMN [20], Petri nets [21], and eEPC [22].

Hybrid. Additionally, there are process modeling notations that combine imperative and declarative modeling [11]. While BPMN 2.0 allows modeling ad-hoc subprocesses [20], BPMN-D facilitates declarative sections in imperative BPMN models [10]. Van der Aalst suggests that accepting Petri nets [21] can include declarative semantics as well. The workflow engine Camunda facilitates interchangeability between CMMN, DMN and BPMN.[1]

3 Related Work

We classify our approach as a workflow execution system specifically designed for flexible processes which allows for modeling routine tasks as subprocesses and delivers a shared context for case data management. In the following, we distinguish our approach from existing ACMS, and BPMS which support knowledge work and routine tasks at the same time. For the identification of related work, we conducted a literature search on *Google Scholar* and examined pertinent conference proceedings, i. e. BPM, CAiSE, ECIS, ICIS, and WI.

Künzle & Reichert introduced PHILharmonicFlows [23], [24] as an object-aware process management framework that focuses on the *data object* as their primary process element. For example, HR processes have an open position and several applicants. Traditional BPM approaches typically have to focus on either the position or the applicant as the process instance, i. e. the primary focus. Here, each data object has its own state. PHILharmonicFlows differentiates micro and macro processes for object behavior and

[1] `https://www.camunda.com` (visited on 2020-05-31)

object interactions [25], [23]. For ad-hoc activities in knowledge work, process support is restricted to predefined (note) attributes to be filled with arbitrary values. Our approach is similar in regard to allowing to manage different subprocess instances that have their own state and can focus on different data objects – or stakeholders in the context of HR. In addition, we allow combining subprocesses with ad-hoc activities and subprocess models for different aspects of the work. Furthermore, we expect that knowledge workers may create their own subprocess models as they can capture single aspects of the work, i. e. they do not need to model the whole process.

Fragment-based production case management [26], [27] divides complex processes with many possible execution paths into shorter process fragments. These describe one particular aspect of the process and are linked with data objects and common activities. Pre and post conditions of activities describe the data flow of a process. The approach allows modeling the happy path, exceptions, and global procedures as individual fragments of a process. Users may choose different variants of fragments, e. g. in contract management, either pre-approved contracts can be applied, or new contracts have to be reviewed. The result of both fragments is an accepted contract that can be a precondition for other fragments. Even though individual fragments can be kept small, modelers have to consider the whole process, and ad-hoc activities and artifacts are not considered in the model. Unlike PHILharmonicFlows, "fragment instances" cannot focus on different data objects, at least not in the sense of multiple data objects of the same type. However, fragment-based modeling can be imported into our approach both as running subprocess instances, but also in regard to dependencies between subprocess instances.

Van der Aalst et al. [28] introduced Proclets, a framework for lightweight speech-act-based interacting workflow processes based on Petri nets. Proclets are Petri nets that are connected via ports and corresponding annotations in regard to cardinality and multiplicity. Individual proclets describe the life cycle of an instance. For the connection, they exchange performatives over channels like email that are stored in a traceable knowledge base. Similar to our approach, individual proclets can capture a different focus on entities as subprocess instances, e. g. for HR. However, all activities for a particular instance are modeled within the Proclet, i. e. it is not intended to further divide the process into smaller aspects of the work. Proclets could also be imported into our approach as a single notation for subprocess instances.

4 Methods

We intend to open adaptive case management to knowledge-intensive processes consisting of structured, semi-structured, and completely ad-hoc activities. Therefore, our approach revolves around two hypotheses:

H1 No predefined process schema is necessary to support traceability in knowledge work.

H2 Routine and ad-hoc activities that share the same context can be supported within the same system of record.

With a speech-act-based approach, Hypothesis H1 has already been investigated [29], and yielded the prototype Agora [30] as well as the spin-off Pertuniti [31], [32]. For Hypothesis H2, a focus on speech acts yielded first results that depend on explicit and implicit (e. g. annotations in process models) documentation of interactions. Here, we focus on H2 without requiring additional annotations. From H2, we can directly derive two research questions:

RQ1 Can routine and ad-hoc activities that share the same context be performed within the same system?

RQ2 Can different execution semantics of routine work be performed within the same system?

For communication-centric knowledge-intensive work, RQ1 can already be affirmed by the prototype Agora [30]. To derive a general answer, we want to extend an approach for process support for completely ad-hoc activities with process support for routine work. In the commercial ACMS Pertuniti[2], no process schema is necessary to perform collaborative work in a traceable way. Prior to our extensions, Pertuniti did not include any workflow engine, and automation was restricted to document generation as well as a REST interface to attach web services to. First, we extend Pertuniti with a workflow engine for a subset of BPMN that performs subprocesses within the same case as ad-hoc activities.

For RQ2, we add additional workflow engines and abstract execution commonalities and differences (see Section 6.5). We affirm RQ2 by providing a case and subprocess model that is not tailored to a specific process modeling notation, and that allows performing subprocess instances of different process modeling notations and execution semantics within the same shared context, i. e. in the same case.

We further evaluate RQ1 and RQ2 by applying this approach in the ACMS Pertuniti, and by demonstrating that it can fulfill the requirements of the following motivating example.

5 Motivating Example

We motivate subprocess modelling in ACM by applying the lecture module *Process Analytics* (PA) as a case, or rather, a set of cases. We consider every term of the lecture PA as one case. It shares a common context and consists of weekly lectures, an excursion, a project, a written examination and other ad-hoc tasks. Although the project differs every year, it comprises a mandatory excursion to an industry partner. The lecture-unit preparation is knowledge-intensive. Some lecture units require revision for the new term. Other units, such as fundamentals, might remain similar for years. Holding lectures is predictable up to a certain degree, but may vary depending on the students' number and participation. For these reasons, it seems unsuitable to model designing and holding lectures.

[2] `https://www.pertuniti.com` (visited on 2020-11-30)

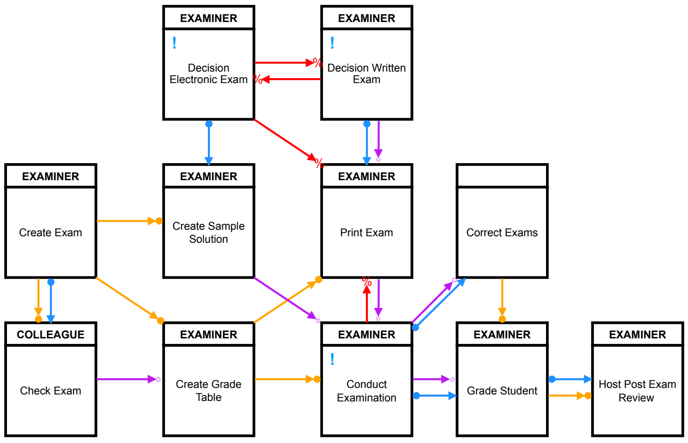


Figure 1. DCR graph: exam preparation

In contrast, exam preparations do not change over the years. They mainly rely on dependency-related information, i. e. we apply declarative modeling. Hence, we implemented a DCR graph to prepare exams (cf. Figure 1). Examiners may decide to conduct the examination orally. In this case, they do not initiate the model. After an examiner has created an exam, it needs a review by a colleague. In declarative notations, case workers can execute activities multiple times when there is no related constraint. In most cases, the examiner creates more than one version of an exam. Before conducting the examination, the examiner must create a grade table. After the examination, the examiner, colleagues, and assistants correct the exams. Thereafter, the examiner enters the grade for each student. Last, examiners must host a post-exam review. The examiner has one decision of whether s/he conducts an electronic or written examination. While electronic exams must include the sample solution, written exams require printing prior to the examination. The single system of record aids in creating the sample solutions, since all lecture slides and transcripts are immediately available.

Planning an excursion is a predictable process. Thus, we modeled organizing it in BPMN (cf. Figure 2). Depending on the industry partner, the responsible needs to enter the date and target location of the excursion in a form. Afterwards, the approval from the central university must be retrieved. If the trip is approved, all students in the project are automatically registered for the excursion. Meanwhile, the organizer compares travel options. If the excursion takes more than a day, the organizer has to search for accommodation. Capturing these options may result in unstructured documents. After the trip, all invoices and bills need to be captured to settle payment.

As we show with the lecture module, one case can comprise structured, semi-structured, and ad-hoc work. Hence, ACM can still benefit from modeled sub processes for routine aspects. We applied notations for routine work based on the process to be supported, which is not supported in available ACM systems.

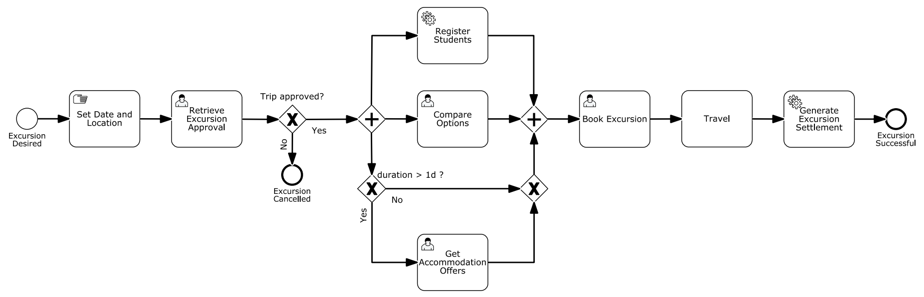


Figure 2. BPMN model: lecture excursion

While the motivating example is academic, this mode of work can be found in many – maybe unexpected – areas: In regulated domains, e. g. medical quality management, core processes have to be documented in advance, and traceability of process instances is mandatory.

6 Notation-agnostic Subprocess Modeling

Our approach to process support enables combining complete ad-hoc activities and modeled processes within the same context, a case. Our approach comprises a case model, dynamic case and process states, and a data dependency model. Furthermore, we outline how an ACMS could handle common and differentiating execution semantics.

6.1 Case Model

Since knowledge-intensive processes consist of ad-hoc and routine fragments, one process model typically cannot capture them. Nonetheless, they should be performed within one traceable context. While concrete artifacts may vary between domains, this section introduces a case model that allows execution of modeled routine subprocesses as well as ad-hoc activities.

Definition 1 (Case Instance). *A case instance $c = \langle s, A, D, I, \Sigma \rangle$ consists of a state $s \in S$ of state types S, a set of master data attributes $A \in K \times V$ as key-value pairs with unique keys, a set of arbitrary artifact data D, a set of subprocess instances I, and a set of performed activities, i. e. an event log Σ.*

Optionally, a case should also contain an unique identifier, and depending on the domain, it may contain a case type of some sort. The set of arbitrary artifact data *D* substantially simplifies actual case management systems. In reality, *D* may contain artifacts of arbitrary types that are supported in different ways, e. g. a calendar for events, a Kanban board for task lists, or some file hierarchy for documents. As many case management systems provide functions to manage case master data, we apply key-value pairs. These can be further specified with e. g. case types. Σ is a consolidated event log of all modeled and ad-hoc activities performed within the case instance.

Definition 2 (Subprocess Instance). *A subprocess instance* $i = \langle \mathcal{M}, s, A, \psi_{\mathcal{N}}, \Sigma_i \rangle$ *consists of a reference to a specific process model* $\mathcal{M}$ *of modeling notation* $\mathcal{N}$*, a state* $s \in S_{\mathcal{M}}$*, i. e. states that are applicable to process model* $\mathcal{M}$*, arbitrary attributes* $A \in K \times V$ *for an internal variable scope if necessary, a notation-specific execution state* $\psi_{\mathcal{N}}$*, and an instance-specific event log* Σ_i*.*

This definition contains three representations of states to capture a variety of requirements. Process states $s \in S_{\mathcal{M}}$ might be useful to derive the success of subprocess instances, e. g. labels on BPMN end events. Knowledge workers may derive the course of action based on states of past subprocess instances, i. e. ACM systems should depict this state prominently. As execution semantics of different notations require different data models, $\psi_{\mathcal{N}}$ can capture notation-specific execution states, e. g. tokens or included activities. Some arbitrary attributes $s \in A$ might be adaptable by end users, otherwise they could also be stored in $\psi_{\mathcal{N}}$.

A **subprocess model** may reference and adapt not only their instance state, but also the case state, e. g. for creating case artifacts or to use case master data at decision points. Hence, subprocess models could be applied as case templates.

6.2 Sketch

Based on the running example, the difference in the context of ACM becomes apparent. Figure 3 depicts the ad-hoc and routine activities involved in giving a lecture on process mining. The case contains many documents and groupware artifacts, e. g. lecture notes and test data, exercises, a course-specific calendar of the lecturer, ad-hoc tasks to improve the lecture over time, and important interactions with stakeholders. The tasks are not modeled in advance as in this case, planning is clearly part of the work and tasks of this granularity are performed once.

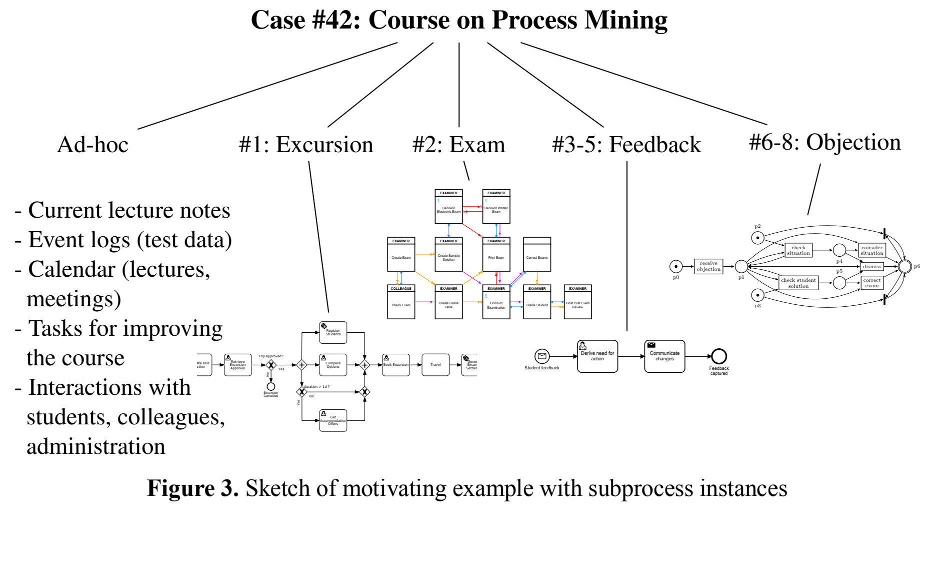


Figure 3. Sketch of motivating example with subprocess instances

Each semester, and not only for this course, students attend excursions to partners of the chair for a guest lecture. Since this aspect is performed sufficiently often, it can be modeled as a structured subprocess model (cf. Figure 2) to facilitate coordination and applying best practices. The same is true for preparing an exam (cf. Figure 1). Processes for feedback and handling objections could also be automated in regard to their documentation and to trigger manual tasks.

The case provides a shared context for the whole lecture. Each modeled process actually is a subprocess, and no redundant data entry for capturing the context is necessary. If one subprocess instance finishes, the whole course may continue until the lecturer no longer expects feedback and objections. Moreover, models for excursions and feedback might be shared with case types for seminars, colloquia, or projects.

6.3 Case and Process State

For ACM, case state primarily coordinates knowledge workers, not workflow engines. Therefore, the typical distinction into [active, finished, aborted] [26], or synonyms like [running, closed, canceled] is neither necessary nor sufficient. Similar to BPMN, knowledge workers may require specific end states where the classification into regularly closing a case and canceling one is not always possible, and maybe additional active states for routing and prioritizing as well. Hence, in [29] and in Pertuniti, these are configurable and can be used for filtering.

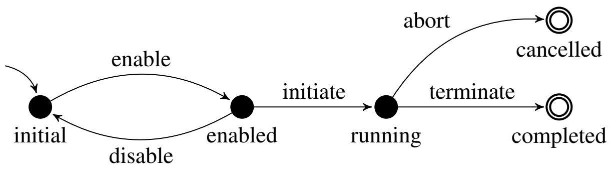


Figure 4. Execution lifecycle of a subprocess instance

Process state coordinates both knowledge workers and workflow engines. Hence, the execution lifecycle is similar to process fragments in Chimera [26], as depicted in Figure 4. If the applied modeling notation provides some sort of end state, e. g. BPMN end events or accepting nodes in Petri nets, the subprocess instance is annotated with this end state as well.

6.4 Data Dependencies

As introduced in Definition 1, the case contains all resulting data artifacts. A subprocess instance should reference external artifacts it depends on either in its attributes A, i. e. V may include reference types, or in the execution state $\psi_{\mathcal{N}}$. If artifacts are stored within the same context, other subprocess instances may use them. For example, in regulated branches, creating a document may trigger reviewing and approving it. In a shared context, the subprocess model does not necessarily have to know the reviewing process, i. e. this approach facilitates loose coupling.

If subprocess models of the same overall case do not know each other, they might impose redundant data entry on knowledge workers. For loose coupling, we can formulate user tasks that optionally request information if necessary: *require*(*type, scope, name*), e. g. *require*("text", case, "location"). Types can be attributes or other artifacts available at the selected scope.

6.5 Common and Differentiating Execution Semantics

While implementing the approach, commonalities and differences between execution semantics become apparent. For typical paradigms, a workflow engine can provide an event log of all activities performed within the instance. Regardless of notation, a workflow engine can derive applicable next activities, e. g. current tokens in BPMN, available tasks in DCR graphs based on markings, and applicable transitions in Petri nets. Depending on the notation, some decisions are performed by the user, some by a workflow engine. Activities can be manual, i. e. the model supports in regard to coordination, or automated.

Differences arise in regard to decisions: Some notations entail that automated activities can be triggered without supervision, some notations require user input as decisions are non-deterministic. Obviously, declarative notations require more sophisticated internal states to capture intermediate results for dependencies. Even token-based notations can be implemented differently, as in BPMN it is relevant which incoming sequence flow has been passed by a token, while e. g. in Petri nets, the amount of tokens suffices. Obviously, these characteristics can be translated to some degree with additional nodes. All notations that we apply in Section 7 require different internal states for execution.

7 Demonstration: Lecture "Process Analytics"

We prototypically implemented the processes outlined in Section 5 and extended the ACMS Pertuniti in regard to offering subprocess workflow engines and enabling execution of multiple subprocesses of different notations. Pertuniti [31], [32] is an academic spin-off and the results of this paper were designed and implemented in close collaboration. It targets knowledge-intensive processes with an emphasis on ad-hoc processes, i. e. when the actual process unfolds as it is performed. No defined process model is the expected default. Pertuniti shows characteristics of groupware and social software, and implements process management as project and knowledge management.

Still, knowledge workers also perform routine work to some degree, and organizations of regulated domains, like healthcare, have to model some processes or aspects in advance. Process models in Pertuniti are implemented according to the outlined approach to combine flexibility of ACM with structure and automation. Processes can be modeled as a subset of BPMN, DCR graphs, and accepting Petri nets. While BPMN and DCR graphs are expected to be applied by customers, accepting Petri nets are primarily intended to show that the approach allows different types of imperative and declarative notations.

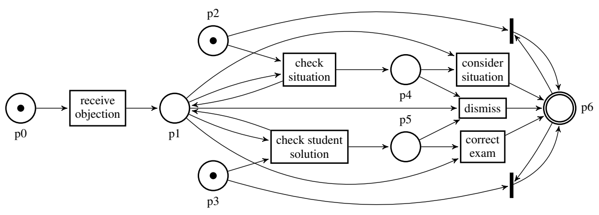
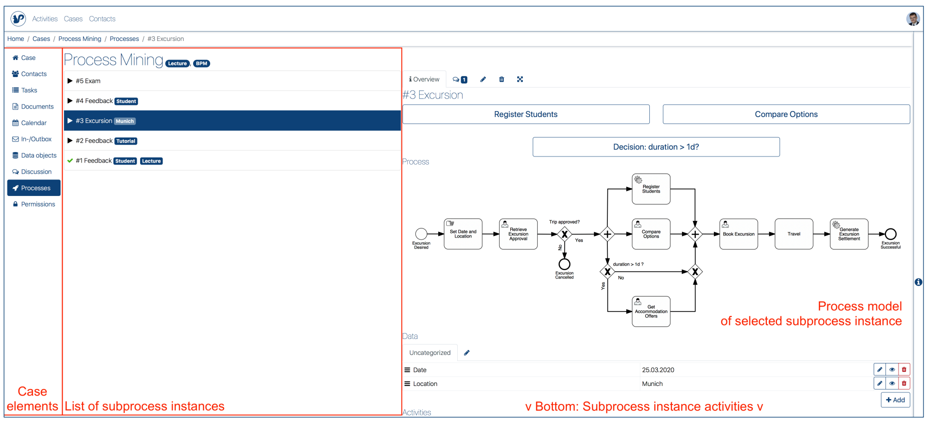


Figure 5. Objections as accepting Petri nets

In Pertuniti, the whole sketch of Figure 3 can be performed in practice. For ad-hoc activities, document management is available for current lecture notes, test data, or other files, and it does not require predefined process models. The course schedule and individual meetings with students and research assistants can be managed in a calendar. Ad-hoc tasks can be managed in categorized lists and a Kanban board. Similar to customer relationship management systems, all additional important communication, e. g. notes on conversations with the exam office or questions that might be good candidates for the exam, can be captured as typed interactions. As all case artifacts are stored within the same system of record, they can easily be directly referenced and commented. Each case and artifact can be annotated with arbitrary attributes in an EAV schema. All activities performed within a case are captured within an "activity stream", i. e. an event log that is displayed to knowledge workers. Activity streams are available on case, subprocess and case artifact level to support traceability and coordination of case workers involved.

Figure 1 for conducting exams can be applied without any adaptations. It is primarily intended for coordination, i. e. the steps are not automated. The service task "Generate Excursion Settlement" of Figure 2 has to be annotated with the appropriate document template and a variable mapping. Register students is currently a manual task. Figure 5 provides an example for an accepting Petri net for objections after the exam. It requires that objections that are dismissed are checked in regard to the situation and solution. For acceptable objections, checking what has been objected to suffices.



Date	25.03.2020
Location	Munich

Figure 6. Running subprocess instances in Pertuniti

All these process models of different notations can be applied as subprocesses within the same context. Figure 6 depicts how these subprocesses are integrated into a case: Each case has a list of subprocess instances. From this list, subprocess models can be initiated, and manual activities and decisions applicable for running instances can be performed. An instance overview depicts the corresponding model and provides an instance-level activity stream.

8 Discussion

We discuss our approach in regard to notions of the term adaptive case management, applicability of different modeling notations within the same system, roles and impact on execution semantics, a case as the shared context of all corresponding activities, and in regard to process mining techniques.

8.1 Adaptive Case Management

Due to the prevalence of model-based ACM systems, Keith Swenson wrote a position paper that postulated "any work support system that depends upon processes designed with BPMN (or BPMN-like languages) cannot be considered an ACM system" [18]. While his statement is intended to raise discussions, we need to show that our approach that combines ACM with BPMN-like languages can still be considered ACM. This question should first be divided into 1) what can be considered as adaptive case management, and 2) whether our approach still fits the definition.

To resolve 1), there are different opinions: CMMN [13] introduces the general concepts with "Any individual `Case` may be resolved in a completely ad-hoc manner, but as experience grows in resolving similar `Cases` over time, a set of common practices and responses can be defined for managing `Cases` in a more rigorous and repeatable manner. This becomes the practice of `Case` management [...]" [13, p. 5], i. e. that case work can be performed completely ad-hoc, but that they see case management only *after* introducing cases in a more rigorous and repeatable manner, or rather by being modeled. They emphasize this point with "A Case has two distinct phases, the design-time phase and the run-time phase" [13, p. 6]. For *adaptive* case management, even the actions to be performed in a case are typically not known in advance [4]. An ACMS has to support knowledge workers in *planning* their course of action, but also in changing the plan [4], and knowledge work unfolds [5]. In knowledge work, planning and performing work converge, i. e. the two *distinct* phases of design-time and run-time should no longer be distinct for *adaptive* case management.

Correspondingly, we apply the term "ad-hoc activity" as actually not being known in advance, even though notations like BPMN contain abstract "ad-hoc tasks". Analogous, modeled process fragments are either structured or semi-structured, but not unstructured work, i. e. we apply "unstructured" similar to "not (yet) modeled". By supporting unstructured work with these definitions, ACM systems could be interesting in project management situations as well.

To answer part 2), our approach does not rely on structured processes, and no process schema is necessary to perform work. Knowledge workers decide whether to apply a process model or perform activities manually. Our approach allows for user-defined routine subprocesses that do not need to be designed by experts, can initially be implemented with little investment and a varying degree of BPM experience, and the concrete implementation facilitates reducing tacit knowledge in process models. Subprocess models act as a support mechanism to reduce manual work and decision making. Work support systems may include means to simplify routine work, even in ACM. As Pertuniti (cf. Section 7) and Agora [29], [30] show, ACM systems can provide structured routine fragments of processes alongside ad-hoc activities.

8.2 Notation-agnostic Process Modeling

Typically, deciding on which modeling notation to use, and whether to apply an imperative or declarative paradigm, entails the choice of the process support system. The appropriate notation highly depends on the process to be supported, and on previous experience of the modeler. Currently, we cannot identify a modeling notation or paradigm that fits every situation or requirement, and most likely, there is none. A notation-agnostic approach of subprocess models with a shared context facilitates modelers to apply the notation they know and want to use. Declarative and imperative paradigms and notations can be combined within the same process support system.

If knowledge workers want to use a subprocess model, they must explicitly invoke it. The knowledge worker decides which activities are necessary or whether a model should be used at all. Changes of the process do not require changing the model as knowledge workers can transparently deviate from it. Changing the model can be performed as soon as the manual effort of deviating from it would be more expensive. Further, in regulated domains, e. g. healthcare, certain processes must be modeled in advance [31]. Such models can be used as subprocess models in our approach. Knowledge workers can then invoke the model to automate particular activities, e. g. generating documents or entering attributes.

Our approach does not require modeling full processes: Routine aspects or fragments suffice to facilitate automation and traceability. By not requiring models for a complete process with all edge cases, end users are enabled to model small routine aspects of their work themselves, and can still continue to manually perform work in the same context. Routine fragments of different types of processes can be combined, e. g. approving and booking travel expenses. Fragments may range from document templates or forms to full process models. Additionally, we prevent modeling processes that may not be reasonable or even feasible for a lot size of 1. A first iteration of support, subsequently, requires only very little investment. As knowledge workers can instantly apply subprocesses, we facilitate them gaining experience and confidence in BPM, which yields better models with less effort.

8.3 Roles and Execution Semantics

Roles in actual knowledge-intensive processes are often fluid. Explicitly capturing them may only be necessary if projects become larger, or in regulated domains. In our approach, roles can be assigned on demand. For example, roles annotated in DCR graphs or BPMN swimlanes can be interpreted as a requirement to be filled. If no user is available for the role, the engine can ask which user should assume it. If knowledge workers want to perform a certain task with an annotated role, the engine can document assuming that role. Once a role is assigned, it is available for other subprocesses as well.

8.4 The Case as Shared Context

A case as the shared context of routine and ad-hoc activities allows reducing manual data entry since subprocess instances may use the same master data and case artifacts. For example, the location of participating at a conference, approving travel expenses and actually booking the trip is the same. Subprocesses may apply data of an instance scope, a case scope, or case artifacts, i. e. not all context is based on global variables. Still, capturing dependencies between context data and subprocesses is an open problem that, with current techniques, does require "programmer-like skills" to fully understand them. Solutions need to find an appropriate compromise that concerns ontologies for coordination, aspects of compiler construction and encapsulation for defining scopes, and a sensible effort for defining and modeling the scope appropriate to non-programming domain experts.

Dependencies between different process models might also become desirable, as subprocesses still *are* aspects of an overall process. As subprocess modeling allows for loose coupling between aspects, capturing dependencies between tasks and subprocesses could be extended to stay loose coupling for concrete artifacts. The speech-act-based approach of Agora [29], [33] is intended to allow just that by deriving inferences and applying rules based on interactions. For dependencies of activities of the same knowledge worker that do not require interactions, approaches like Chimera [26] could be generalized to our approach. This way, one could derive "aspect-oriented BPM" similar to aspect-oriented programming.

8.5 Towards Adaptive Case Mining

Finally, making activity streams explicit on different granularities could facilitate process mining techniques for routine aspects of a case. Currently, a happy path for ACM case instances is not very sensible as knowledge workers can assume a lot size of 1. However, discovery, conformance checking, and enhancement [34] of routine fragments of a case could facilitate automation and execution. For that, process mining techniques might need to be adapted in regard to record linkage and to better correlate and ignore ad-hoc activities and artifacts. Currently, the granularity of subprocess instance event logs allows applying process mining techniques, but not yet in the context of the whole case.

9 Conclusion

We introduced an approach to combine ad-hoc activities and automation in ACM via subprocess modeling. A case serves as the shared context all activities are performed in, and contains all routine and ad-hoc case artifacts. Knowledge workers can transparently deviate from predefined subprocess models. They may add additional tasks in a case, change any case artifacts or deciding if and which subprocess models to invoke. The support system implementing our approach serves as a system of record for the whole case.

We implemented the lecture module process analytics in the ACMS Pertuniti that applies our approach and enables executing subprocesses of cases modeled in BPMN, DCR graphs, and accepting Petri nets. Our motivating example uses all three notations and requires ad-hoc activities as well. The approach makes all ad-hoc and routine activities transparent by providing activity streams on case, subprocess instance, and case artifact level.

Still, some challenges remain. First of all, we plan to conduct an in-depth empirical evaluation of our approach. As it is prototypically implemented in the commercial ACMS Pertuniti, we are going to assess the quality of our approach in a real-world setting. Further topics are open for future research from a technical point of view. Due to the isolation of subprocess instances, dependencies between instances are only captured by the shared context. To remain loosely coupled, solutions might consider further abstractions of goals and tasks. Moreover, data dependencies between models currently are solved similar to global variables in programming languages, and do not consider ontologies and more sophisticated encapsulation. Finally, our focus on event logs of different granularity unveils that process mining for routine aspects of knowledge-intensive processes might become a viable option. Process mining techniques need to be adapted in regard record linkage and correlating ad-hoc artifacts to really make process mining a viable option in ACM.

References

1. Lund, S., Manyika, J., Ramaswamy, S.: Preparing for a new era of knowledge work. McKinsey Quarterly 4, 103–110 (2012)
2. Swenson, K.D.: Robots don't innovate - innovation vs automation in BPM (May 2015)
3. Drucker, P.F.: Knowledge-worker productivity: The biggest challenge. California Management Review 41(2), 79–94 (1999)
4. Swenson, K.D.: Innovative organizations act like systems, not machines. In: Fischer, L. (ed.) Empowering Knowledge Workers: New Ways to Leverage Case Management, pp. 31–42. Future Strategies Inc. (2014)
5. Swenson, K.D.: The nature of knowledge work. In: Mastering The Unpredictable: How Adaptive Case Management Will Revolutionize The Way That Knowledge Workers Get Things Done. Meghan-Kiffer Press (2010)
6. Hildebrandt, T., Marquard, M., Mukkamala, R.R., Slaats, T.: Dynamic condition response graphs for trustworthy adaptive case management. In: LNCS. vol. 8186, pp. 166–171 (2013)

7. Pesic, M., van der Aalst, W.M.P.: A declarative approach for flexible business processes management. Business Process Management Workshops pp. 169 – 180 (2006)
8. Haisjackl, C., Barba, I., Zugal, S., Soffer, P., Hadar, I., Reichert, M., Pinggera, J., Weber, B.: Understanding Declare models: strategies, pitfalls, empirical results. Software & Systems Modeling 15(2), 325–352 (2016)
9. Zugal, S., Soffer, P., Haisjackl, C., Pinggera, J., Reichert, M., Weber, B.: Investigating expressiveness and understandability of hierarchy in declarative business process models. Software & Systems Modeling 14(3), 1081–1103 (2015)
10. de Giacomo, G., Dumas, M., Maggi, F.M., Montali, M.: Declarative process modeling in BPMN. In: LNCS. vol. 9097, pp. 84–100 (2015)
11. Schönig, S., Jablonski, S.: Comparing Declarative Process Modelling Languages from the Organisational Perspective. In: International Conference on Business Process Management, pp. 17–29 (2016)
12. van der Aalst, W.M.P., Pesic, M., Schonenberg, H.: Declarative workflows: Balancing between flexibility and support. Computer Science-Research and Development 23(2), 99–113 (2009)
13. Object Management Group: Case Management Model and Notation (CMMN). Tech. rep. (2016), `https://www.omg.org/spec/CMMN/1.1/PDF`
14. Mukkamala, R.R.: A Formal Model For Declarative Workflows. Phd, IT University of Copenhagen (2012)
15. Zeising, M., Schönig, S., Jablonski, S.: Towards a common platform for the support of routine and agile business processes. In: International Conference on Collaborative Computing: Networking, Applications and Worksharing, pp. 94–103 (2014)
16. de Leoni, M., Maggi, F.M., van der Aalst, W.M.P.: Aligning event logs and declarative process models for conformance checking. In: LNCS. vol. 7481, pp. 82–97 (2012)
17. Fahland, D., Lübke, D., Mendling, J., Reijers, H., Weber, B., Weidlich, M., Zugal, S.: Declarative versus Imperative Process Modeling Languages: The Issue of Understandability. In: Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Soffer, P., Ukor, R. (eds.) Enterprise, Business-Process and Information Systems Modeling. Springer Berlin Heidelberg (2009)
18. Swenson, K.D.: Position: BPMN is incompatible with ACM. In: Rosa, M., Soffer, P. (eds.) Business Process Management Workshops: BPM 2012 International Workshops, Tallinn, Estonia, September 3, 2012. Revised Papers, pp. 55–58. Springer Berlin Heidelberg (2013)
19. van der Aalst, W.M.P.: Process Mining: Data Science in Action. Springer Berlin Heidelberg, Berlin, Heidelberg, Second Edition (2016)
20. Object Management Group: Business Process Model and Notation (BPMN) (2011), `http://www.omg.org/spec/BPMN/2.0`
21. van der Aalst, W.M.P.: Everything You Always Wanted to Know About Petri Nets, but Were Afraid to Ask. In: Hildebrandt, T., van Dongen, B.F., Röglinger, M., Mendling, J. (eds.) Business Process Management. pp. 3–9. Springer International Publishing (2019)
22. Scheer, A.W.: ARIS—vom Geschäftsprozess zum Anwendungssystem. Springer-Verlag (2013)
23. Künzle, V., Reichert, M.: Philharmonicflows: towards a framework for object-aware process management. Journal of Software Maintenance and Evolution: Research and Practice 23(4), 205–244 (2011)
24. Künzle, V., Reichert, M.: Philharmonicflows: research and design methodology. Universität Ulm (2012)
25. Künzle, V., Reichert, M.: A modeling paradigm for integrating processes and data at the micro level pp. 201–215 (2011)

26. Hewelt, M., Weske, M.: A hybrid approach for flexible case modeling and execution. In: La Rosa, M., Loos, P., Pastor, O. (eds.) Business Process Management Forum: BPM Forum 2016, Rio de Janeiro, Brazil, September 18-22, 2016, Proceedings. pp. 38–54. Springer International Publishing, Cham (2016)
27. Meyer, A., Herzberg, N., Puhlmann, F., Weske, M.: Implementation framework for production case management: Modeling and execution. In: Enterprise Distributed Object Computing Conference (EDOC). pp. 190–199 (9 2014)
28. van der Aalst, W.M.P., Barthelmess, P., Ellis, C.A., Wainer, J.: Proclets: A framework for lightweight interacting workflow processes. International Journal of Cooperative Information Systems 10(04), 443–481 (2001)
29. Tenschert, J.: Speech-Act-Based Adaptive Case Management. Ph.D. thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen (2019)
30. Tenschert, J., Lenz, R.: Agora - speech-act-based adaptive case management. In: Azevedo, L., Cabanillas, C. (eds.) Proceedings of the BPM Demo Track 2016 Co-located with the 14th International Conference on Business Process Management (BPM 2016), Rio de Janeiro, Brazil, September 21, 2016. CEUR Workshop Proceedings, vol. 1789, pp. 61–66 (2016)
31. Tenschert, J., Hormesch, M.: Flexibility, adherence, and guidance for regulated processes with case management. In: BPM 2020 Industry Forum, CEUR Workshop Proceedings (2020)
32. Tenschert, J., Marmaridis, S.: Pertuniti: Subprocess modeling and hierarchic case management. In: BPM 2020 Demo Track, CEUR Workshop Proceedings (2020)
33. Tenschert, J.C., Lenz, R.: Towards speech-act-based adaptive case management. In: AdaptiveCM 2016–5th International Workshop on Adaptive Case Management and other non-workflow approaches to BPM (2016)
34. van der Aalst, W.M.P., Adriansyah, A., de Medeiros, A.K.A., Arcieri, F., Baier, T., et al.: Process mining manifesto. In: Business Process Management Workshops. pp. 169–194. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)