Extending case models to capture organizational aspects and time

Anjo Seidel[1,*], Pierre Burghardt[1], Maximilian König[1] and Mathias Weske[1]

[1]*Hasso Plattner Institute, University of Potsdam, Prof.-Dr.-Helmert Str. 2–3, Potsdam, 14482, Germany*

Abstract

In knowledge-intensive processes, knowledge workers need to plan the future execution of their cases, which is constrained by the expected duration of possible activities and the available human resources to perform them. Yet, contemporary knowledge-intensive process modeling approaches do not support both the representation of the expected duration of actions and the organizational aspects of the process. Therefore, this paper proposes an extension of the fragment-based case management approach to capture the required time and available resources for activity executions. We propose a syntactical extension and translational semantics to colored Petri nets. A provided prototypical implementation allows modeling the extended case models.

Keywords

Business Process Management, Knowledge-intensive Processes, fragment-based Case Management, Colored Petri Nets

1. Introduction

Business process management aims to support the processes in organizations by providing process models and techniques to support their execution [1]. Knowledge-intensive processes (KiPs) are a class of business processes that are inherently complex, emergent and require flexible modeling techniques [2]. Different modeling approaches were developed [2, 3], one of which is fragment-based case management (fCM) [4, 5].

One crucial task in knowledge work is planning [6]. That is aligning actions according to a process instance goal [7]. The process model can be of use as it describes the ordering constraints of activities. Approaches were already presented to support the planning of KiPs by transforming such processes into automatic planning problems [8, 9]. For fCM, decision support can be generated by analyzing the state space, i.e., all possible behavior of the process model [10, 11].

Yet, the fCM approach does not support the modeling of all aspects that are crucial for planning. On the one hand, the human resources that are required to execute activities in the process need to be represented in the process model [2]. Second, knowing how long certain

ER2023: Companion Proceedings of the 42nd International Conference on Conceptual Modeling: ER Forum, 7th SCME, Project Exhibitions, Posters and Demos, and Doctoral Consortium, November 06-09, 2023, Lisbon, Portugal

*Corresponding author.

anjo.seidel@hpi.de (A. Seidel); pierre.burghardt@student.hpi.de (P. Burghardt); maximilian.koenig@hpi.de (M. König); mathias.weske@hpi.de (M. Weske)

0000-0002-9652-5340 (A. Seidel); 0000-0002-2244-1179 (M. König); 0000-0002-3346-2442 (M. Weske)

activities will take during planning is important. Resources may be busy with other tasks for a certain time. Current case models lack this perspective and cannot represent these constraints fully. Therefore, planning support is currently only limitedly available.

To overcome the limited representation of organizational aspects and time in case models, this paper proposes an extension of fCM, allowing to model of the required resources and the expected duration to execute activities. Besides a syntactical extension to specify those two aspects, we define translational semantics for the extended case models by extending the existing formal semantics for fCM [4]. An existing modeling tool for fCM [12] is extended to allow modeling the extensions and serve as a foundation for planning support.

The remainder of this paper is structured as follows. Preliminary contributions and definitions are provided in section 2. Next, section 3 introduces the syntactical extensions and their formal semantics, before the prototypical implementation is presented in section 4. The existing work related to this topic is presented in section 5 before section 6 discusses and concludes the paper.

2. Preliminaries

Different streams of work are preliminary to this paper. As a formal definition for process models, colored Petri nets (CPN) are introduced. Furthermore, the knowledge-intensive process modeling approach fragment-based Case Management (fCM), and its current translational semantics to CPNs are elaborated.

2.1. Petri nets and colored Petri nets

Petri nets are directed bipartite graphs that contain places and transitions connected by directed arcs [13]. A distribution of tokens to places is a marking of the net. If all preceding places contain a token, a transition may fire, which will consume a token from each preceding place and produce a token for all succeeding places. They are commonly used to define translational semantics of process modeling languages [14] like Business Process Model and Notation (BPMN) [15].

Colored Petri nets (CPNs) are an extension of regular Petri nets [16]. An example CPN is displayed in Figure 1. In addition to places, transitions, and arcs, they define colorsets for places. As colorsets define attributes and types for tokens, tokens can hold concrete attribute values and can be distinguished. Transitions may define guards that formally express conditions on whether the transition is allowed to fire based on the tokens' values. Arc expressions define the amount and value of tokens consumed or produced by transitions. The initial marking of the net assigns a set of tokens to every place.

The place P_1 in Figure 1 can hold tokens of the colorset $C_1 \subseteq \mathit{String} \times \mathit{Boolean}$, a tuple with a *name* attribute and a *used* attribute. The place P_2 can hold tokens of the type $C_2 \subseteq \mathit{String}$. The transition T_1 requires a token from P_1. The guard checks whether the used attribute is false. The firing of transition T_1 produces a token to P_2 with the same value as the token name and produces a token back to P_1 where the *used* attribute is set to true. Therefore, every token can only be used once by T_1. In the initial state, there are three tokens in P_1. T_1 can fire three times in total, producing three new tokens in P_2.

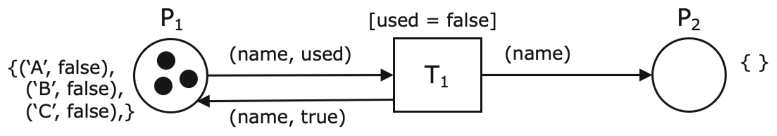


Figure 1: An example colored Petri net with a transition T_1 updating tokens from place P_1 and creating tokens for P_2.

2.2. Fragment-based Case Management

The fCM approach was first introduced by Hewelt et al. [5] and later refined by Haarmann [4]. In general, an fCM model consists of four components: (i) process fragments describing the activities in the process and their ordering and data constraints, (ii) a domain model defining the involved data classes and their relations, (iii) object behavior that specifies the allowed behavior for all data instances, and (iv) a termination condition stating when the case may be terminated. FCM is a hybrid approach that combines procedural control flow with declarative data constraints. The running example of renovating a flat is used to illustrate the concepts.

Process fragments. The process fragments have similar semantics to BPMN [15] consisting of events, activities, and their control flow, while data objects define data constraints for activities and therefore pose declarative data constraints. Fragments can be executed individually, concurrently, and repeatedly, constrained only by the required data objects.

In Figure 2, four fragments define the process of renovating a flat with multiple rooms. As soon as the renovation starts in fragment *F1*, the flat is in the state *in progress*. For the flat that is in progress, the activity *clean room* can be executed in *F2* resulting in a data object *room* in the state *cleaned*. This clean room can then be utilized as input for the fragment *F3* to paint the cleaned room, or for the fragment *F4* to lay tiles. Both activities have two possible inputs. Either, a cleaned room or a room with already laid tiles can be painted in F3. In F4, tiles can also be laid after the room is painted. This allows the execution of the two activities in arbitrary order for a single room. Additionally, each of the two fragments can be executed concurrently for multiple rooms. Finally, once a set of rooms has paint and tiles, the flat can be inspected and either accepted or rejected.

Domain model and object behavior. The fragments define data constraints based on the data classes, which are defined in the domain model. For the flat renovation, it defines the case class *flat* (cf. Figure 3). For each flat, there can be up to five rooms. Each room can have one instance of paint and one instance of tiles.

For each data class, a set of states and allowed state transitions are defined in the object behavior. Data object states are abstractions from the concrete attribute values. In Figure 4, flats are defined to be in progress before being either accepted or rejected. Rooms can be cleaned, then either be painted or have tiles laid before having both. Paint can be in the state dry and Tiles can be laid. The fragments can use these defined states but must comply with the defined state transitions of the data objects.

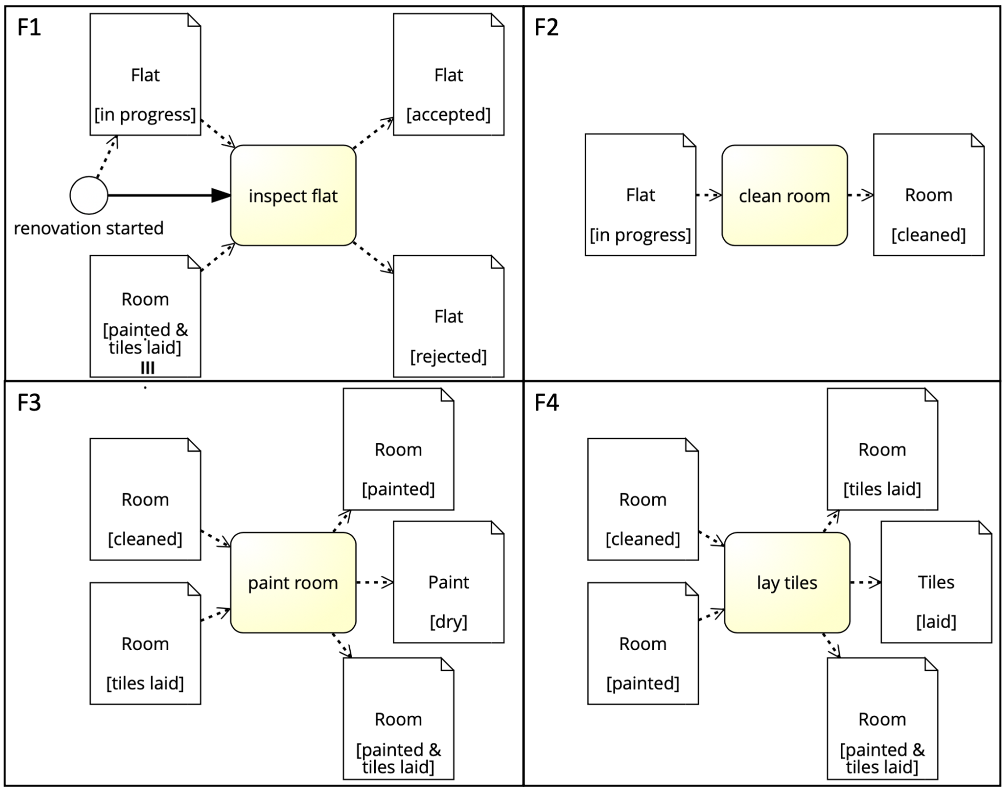


Figure 2: The process fragments describing the activities and their ordering constraints for a renovation process.

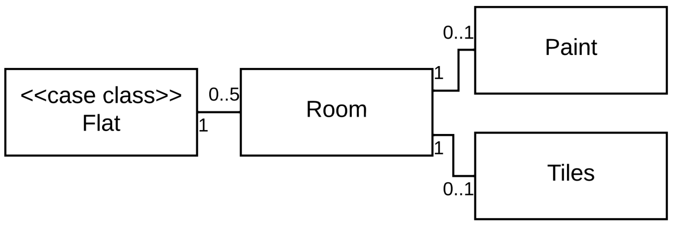


Figure 3: The domain model of the renovation process as a UML class diagram.

Translational semantics. The most recent formal semantics for fCM are defined by Haarmann [4] by providing a mapping of fCM to CPNs. Due to the complexity of the original mapping, we reduce the introduction to the most relevant concepts and those that will be extended in the following.

In the mapping, each possible output of an activity is represented as one transition. The CPN in Figure 5 represents the activity *paint room* in fragment F3 for the input of a cleaned room and the respective output of a painted room. Painting a room with tiles is represented by a second transition. The colorsets of the resulting CPN consist of a colorset for control flow tokens and a colorset for each data class. In order to allow for identifying and correlating data

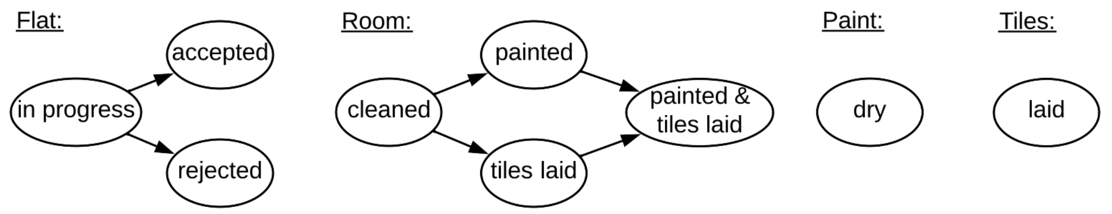


Figure 4: Object behavior defined by a state transition system for each data class.

objects, Haarmann defines tokens to have an ID as attribute [4]. Every control flow arc in the fragments is represented as a place. Also, every defined state of every data class is assigned a place. Data object instances in the current execution state are represented as tokens on the according places. In the current state, the transition for painting the room may fire for the cleaned room *r2* or *r3*. The firing of the transition consumes this token and produces one data object instance of the according room on the place for painted rooms, and a token for dry paint. Afterward, the other transitions in the full mapping can use the painted room token. If multiple tokens allow for different executions of one transition, they can be used concurrently, i.e., the rooms *r2* and *r3* can be painted in arbitrary order.

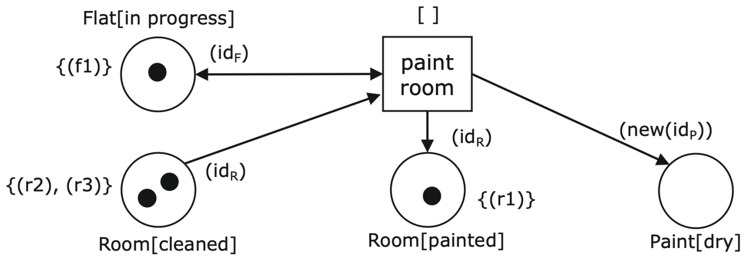


Figure 5: The reduced mapping of *paint room* for a cleaned room as input according to Haarmann [4].

3. Extending case models with roles, resources, and time

In order to provide knowledge workers with assistance in planning their process ahead, process models need a notion of human resources that can perform activities and the expected duration. In the following, we first provide a syntactical extension of fCM and, second, translational semantics of these extensions.

3.1. Extensions of fCM

Roles and resources. For the semantics of roles, we follow the semantics of the de facto industry standard BPMN [15], in which roles can be defined as lanes in an organization's pool. A lane comprises multiple activities, which can be executed by resources of this role [1]. Human resources are assumed to possibly belong to multiple roles. In order to capture the utilized roles and the available resources, the case model needs to be extended by a definition for both.

The set of roles *Roles = {FM-Team, Painter, Tiler}* for the renovation process contains facility managers, painters, and tilers.

The resource model describes a set of available resources. As case management allows for changing models during run-time, this set might change during the execution. Each resource is a tuple of a unique identifier $id \in ID$ and a set of roles: $Resources \subseteq ID \times \mathscr{P}(Roles)$. For the renovation example, we assume the following available craftspeople:
$Resources = \{(Anna, \{FM\text{-}Team, Painter\}), (Bert, \{Painter, Tiler\}), (Clara, \{Tiler\})\}$.

Additionally to BPMN, we allow that an activity can require multiple resources of the same role for execution. This capacity is a natural number of resources.

To model the required roles, we propose a syntax that deviates from BPMN. Because large models could become complex if fragments expand over multiple lanes, the role and the capacity for an activity are defined as a prefix to the activity label. As displayed in Figure 6, the activity *paint room* receives the label *"Painter (2):"*. To execute this activity, two resources with the role *Painter* are required. Anna and Bert must both be available to execute the activity.

If no role and no capacity are defined for an activity, its semantics are not affected. It can be executed without additional constraints.

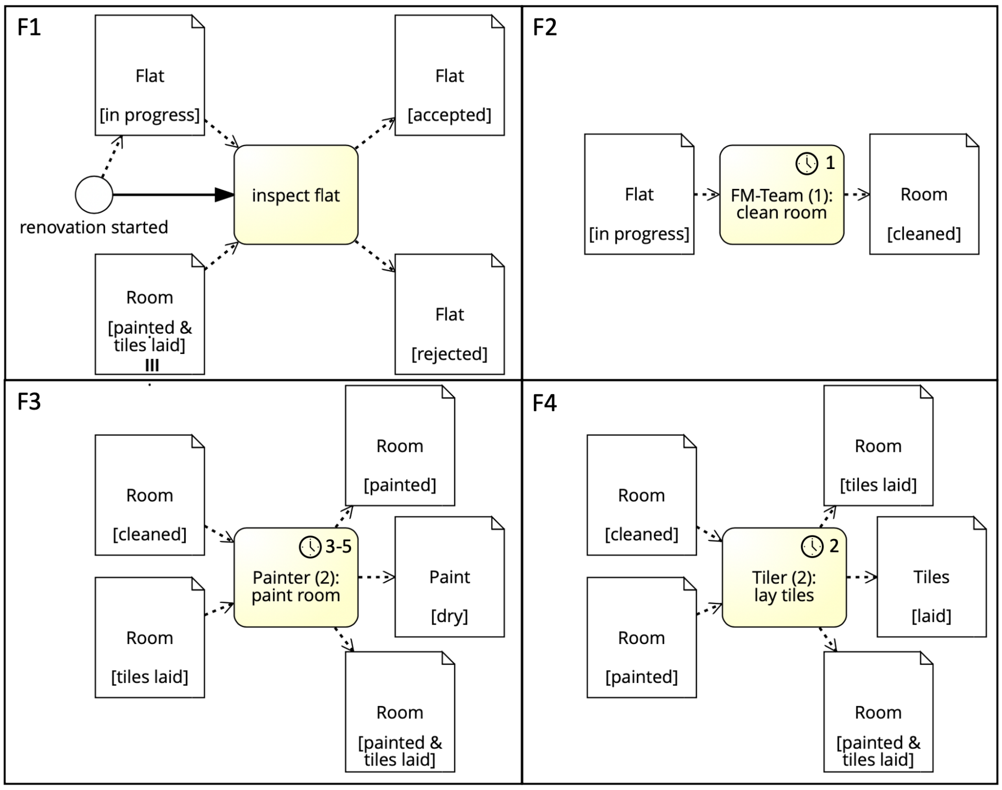


Figure 6: Extended fragments, where activities define required roles and capacities, and the expected duration for the renovation process.

Time. In order to plan the future execution of the process, it needs to be specified how long it takes for certain activities to be executed. The execution of an activity may keep resources busy and cannot be assigned to other activities in parallel. Inspired by the work of Eder et al. [17], we propose to denote the expected duration for an activity as a tuple of a minimal duration and a maximal duration. It has a lower bound l and an upper bound u.

As an abstraction for the concrete domains of the case models, the duration is specified in time units. In the use case of renovating a flat, a suitable time unit may be a working day. The expected time units required to execute activities can be denoted as parameterizable activity modifiers as illustrated in Figure 6. Painting a room is expected to take 3 to 5 working days while laying tiles is certain to take 2 working days.

3.2. Translational semantics

We provide the translational semantics to formally define the previously discussed aspects. The semantics require a notion of running activities and concurrent access to data, a representation of roles and resources, and a notion of time.

Running activities. For the use case of planning, a distinction between the beginning and the termination of an activity is required. For BPMN, this is already defined to map, for instance, boundary events to Petri nets [14]. As illustrated in Figure 7, activities are represented by a begin transition that depicts the starting of the activity. It produces a token to a running place, which can be consumed by a terminate transition representing the termination of the activity. The colorsets of the CPN are extended by the colorset *Running*. Every running place with this colorset holds tokens that represent the activity instances that are currently running. In order to correlate the correct data objects after termination, the running token holds a list of the data object identifiers that it is executed on. This is set by the begin transition and the terminate transition changes exactly those data objects.

Handling parallel data access. Introducing the running place for activities poses an important challenge. If two activities can take the same data object as input but change it in a different way, then they must not be allowed to edit it at the same time. While the painting of a room is currently in progress, the data object for this room must not be used as input for laying tiles. The token for the room must be locked from accessing it.

We propose a locking mechanism that locks those data objects that will be changed during the execution. This is represented in the CPN by adding an attribute to the data classes' color set. This *locked* attribute is set to true if the data object is currently subject to changes. If it is false, a data object can be used by activities. All data objects need to be unlocked to use them as a pre-condition. Data objects that are only read, do not have to be locked during the execution.

In the CPN (cf. Figure 7), the begin transition's guard needs to check whether the used data object tokens are locked. The begin transition sets the room's locking attribute to true, while the terminate transition sets it to false again. Afterward, the data object can be used again.

Process resources. To provide a notion of the available resources and also those resources that are currently busy with executing activities, a new colorset *Resources* is introduced. Also,

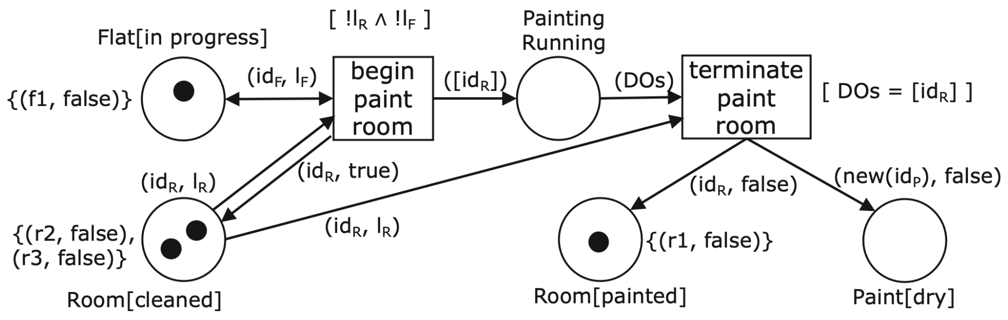


Figure 7: CPN representation of *paint room* with a *Painting Running* place and data locking.

two places for this colorset are added to the CPN. One for the available and a second for the busy resources. The colorset defines tokens representing resources as introduced in subsection 3.1. As seen in Figure 8, present resources are depicted as tokens either on the available or the busy place. Once painting is started by firing $paintroom_b$, the human resources Anna and Bert, who are able to paint, are set to busy. When the terminate transition for *paint room* fires, the resources are put back into the place for available resources.

In order to correlate those resources that perform an activity instance together, the running token holds a list of the executing resources' identifiers.

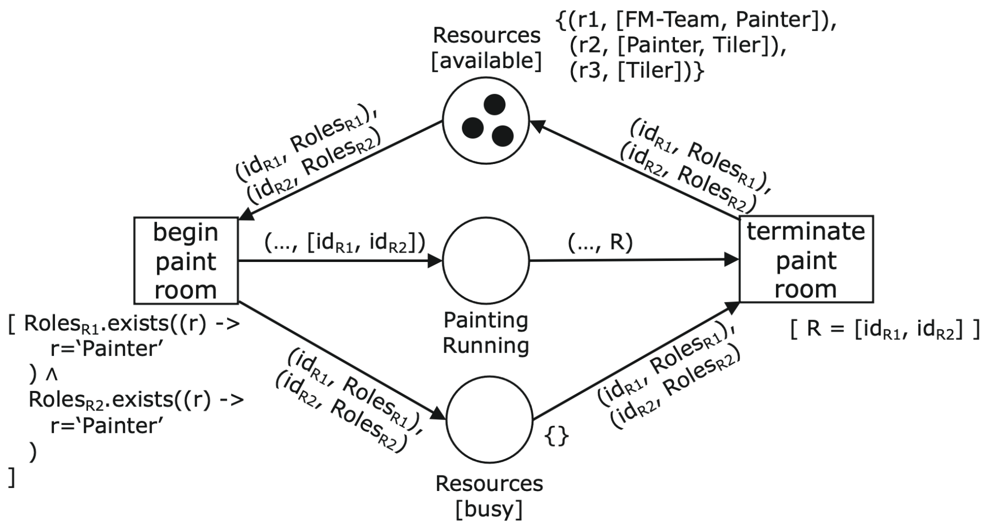

Resources [available]
{(r1, [FM-Team, Painter]), (r2, [Painter, Tiler]), (r3, [Tiler])}
(id_R1, Roles_R1), (id_R2, Roles_R2)
(id_R1, Roles_R1), (id_R2, Roles_R2)
begin paint room
(..., [id_R1, id_R2])
(..., R)
terminate paint room
[Roles_R1.exists((r) -> r='Painter') ∧ Roles_R2.exists((r) -> r='Painter')]
Painting Running
[R = [id_R1, id_R2]]
(id_R1, Roles_R1), (id_R2, Roles_R2)
(id_R1, Roles_R1), (id_R2, Roles_R2)
{}
Resources [busy]

Figure 8: CPN representation of *paint room* with a *Painting Running* and places for available and busy resources.

Passing of time. Time can be represented in the CPN by introducing a new colorset *Time*. It holds exactly one time token with a natural number for the current time stamp. It is initialized with zero and can be incremented by a newly introduced clock transition, which consumes a time token and increments its value by one. The firing of the clock represents the passing of one time unit in the case model.

To make sure that an activity runs for a certain amount of time, each running token is extended with the lower bound *l* and the upper bound *u*. The begin transition sets both according to the defined activity and relative to the current time stamp. The *l* holds the earliest and *u* the latest point in time when the activity is allowed to terminate. The terminate transition is only enabled if the current time token is in the interval of $[l, u]$. For painting, given the current time stamp 2, *l* is set to 5 and *u* to 7 (cf. Figure 9).

Still, the problem arises that the time could be incremented over the *u* of an activity instance, preventing it from terminating. Therefore, the clock must not fire as long as any terminate transition is required to fire. This can be assured by introducing a new colorset and place for running actions, i.e., activity instances. This place holds a token with a list of tuples. Each tuple contains the name of the currently running action and its relative *l* and *u*. The begin transition will add the action to the list and the terminate transition removes it. With this set of currently running actions, the clock can check in its guard whether running actions have to terminate first. If yes, it is disabled until the respective termination transition is fired.

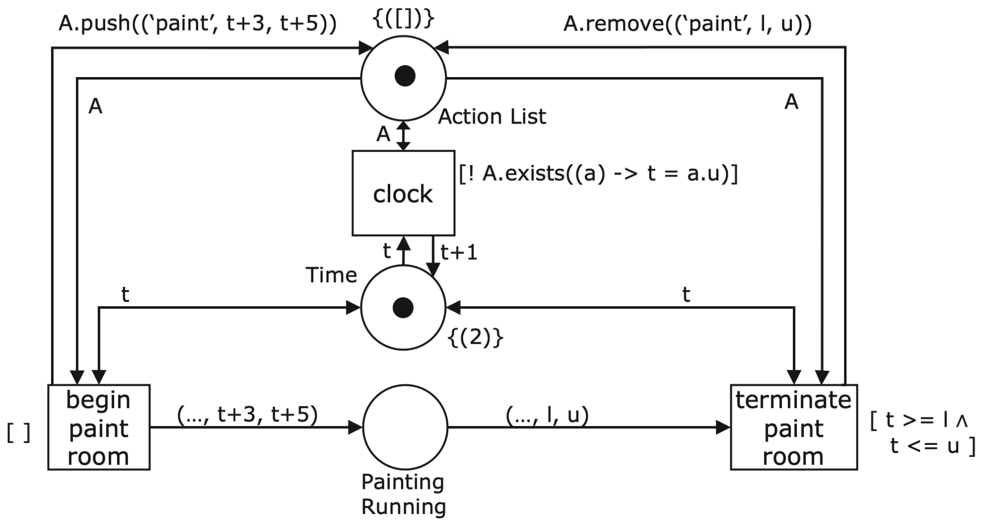


Figure 9: CPN representation of *paint room* with, the *Time* place and the *Action List* place to represent the passing of time.

Complete mapping. The explained and illustrated extensions can be applied to a CPN resulting from the mapping of Haarmann [4]. Of course, only syntactically extended activities need to be mapped accordingly. For instance, inspecting the flat does not require any resources and does not take any time in the model. It is not extended in the CPN.

To ensure that all activities have the same notion of time and the same pool of resources, one global time place, clock transition, and resource places are defined for the case model. In Figure 10, a schematic mapping of the whole case model is visualized. The semantics were extended by introducing a running place for activities that need time to execute, places for available and busy resources, and a time place with a clock and the action list.

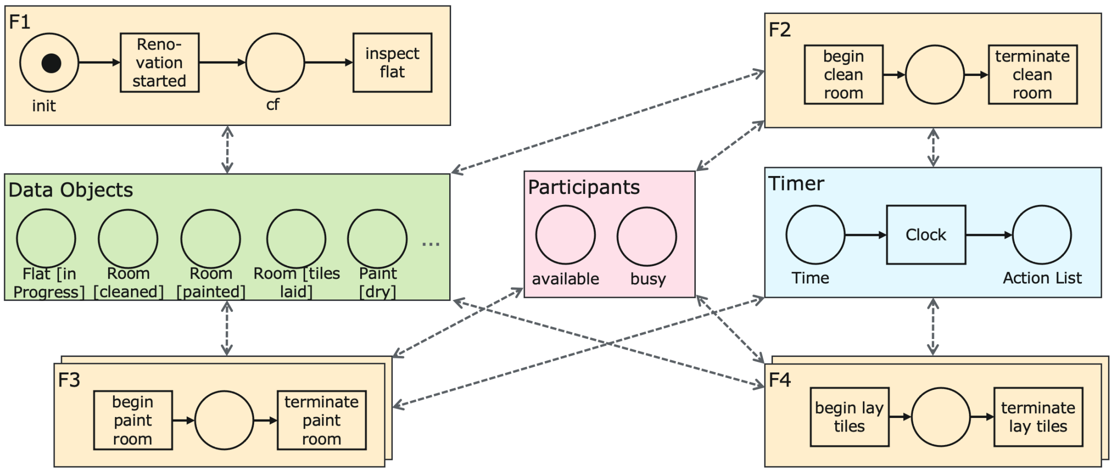
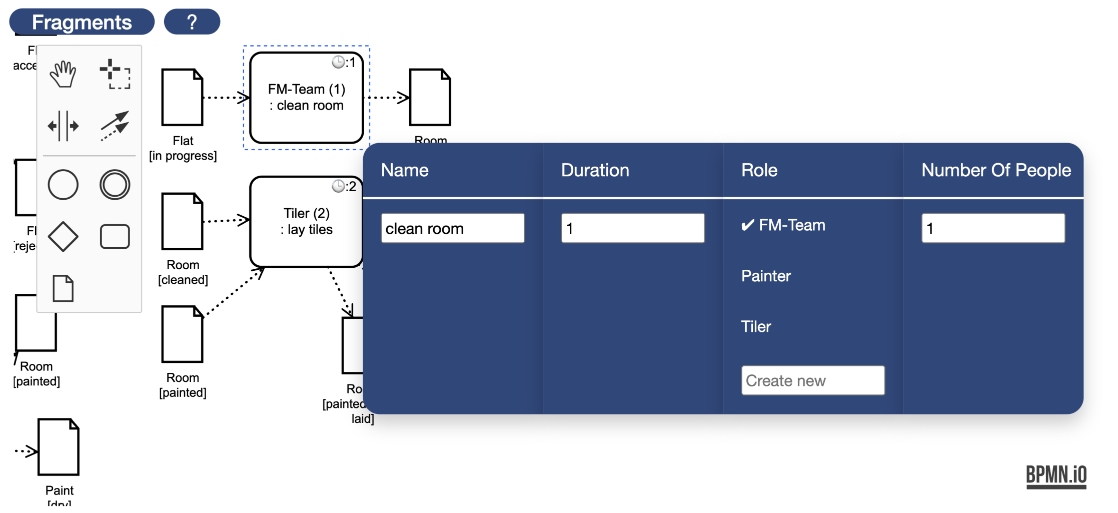


Figure 10: Schematic visualization of extended CPN representation with running places for running activities, global resource places, and the global time representation.

4. Prototypical implementation

In this chapter, we provide an enhanced web-based modeling tool for fCM[1]. It is based on fcm-js [12], which allows consistent modeling of fCM case models that consist of fragments, a domain model, object behavior, and a termination condition. Our prototype extends the fCM case modeler with a role modeler and a resource modeler, and inputs for roles and time in the fragment modeler as elaborated in Section 3.1.



Name	Duration	Role	Number Of People
clean room	1	✔ FM-Team	1
		Painter	
		Tiler	
		Create new	

Figure 11: The fragment modeler allows modeling activities with a specification of the required roles and capacities and the expected duration.

[1]Source and documentation: https://github.com/bptlab/fCM-design-support/tree/planning-support

Figure 11 shows the enhanced fragment modeler of our prototype on our example of the flat renovation. The activities' syntax is extended with duration and resource requirements in terms of capacity and their role, following the illustration in Figure 6. A double-click on an activity opens the depicted menu allowing for configuration of its name, expected duration, executing role, and required capacity.

A separate role modeler allows for modeling the roles that can be selected for activities. The resource modeler allows the definition of the case's resources, to which the defined roles can be assigned. The prototype ensures consistency between the different modelers and within the fragments. Only positive durations are allowed, and only roles that are actually defined can be referenced in the fragment and resource modeler.

5. Related work

The work that is related to this paper consists of the definition of KiPs, KiP modeling approaches, organizational modeling in BPM, and the use of time in process models in general and in Petri nets in particular.

Di Ciccio et al. define and characterize KiPs and derive modeling requirements, for instance, modeling knowledge workers and their roles in the process [2]. Pyoria et al. emphasize planning as an important knowledge-intensive task [6]. Also, knowledge workers, i.e., human resources, collaborate in their planning in order to reach their individual and collective goals [18].

Besides case management, different modeling approaches exist that aim to model KiPs. Di Ciccio et al. [2] and Stainau et al. [3] reviewed contemporary approaches like Declare, Guard-Stage-Milestone, DCR-Graphs, or OCBC, which use declarative, data-centric, or hybrid approaches. The Case Management Model and Notation [19] emerged and fragment-based Case Management was proposed [4, 5]. In prior work, fCM models were shown to be useful to automatically provide knowledge workers with model-driven decision support [11].

Even though the modeling of human resources in business process management is understudied, the concept of roles is widespread. Zur Mühlen [20] describes a meta-model for roles and resources as well as resource allocation patterns. Process modeling techniques like BPMN [15] allow for modeling roles as swimlanes, while resources may belong to multiple roles in the organization. Declare defines roles that can be assigned to concrete human resources during the instantiation of the process [21]. For fCM, an extension for modeling roles and resources was already proposed but not semantically defined by Remy [22].

Cheikhrouhou et al. provide an overview of different contributions to the temporal aspects in BPM and emphasize the duration of activities as an important aspect in business process models [23]. For instance, Eder et al. [17] define an extension of workflow systems.

Contemporary Petri net variations exist that can represent time. Popova-Zeugmann [24] defines the following three types: (i) Time Petri nets, in which transitions have a clock and an interval of time units for which they can fire. Yet, transitions can not distinguish different tokens for enablement. (ii) Timed Petri nets are eager, or greedy, in firing. A minimal set of transitions fires as soon as it is enabled, which restricts finding optimal plans. (iii) Petri nets with time windows have a time interval for every token that is in a certain place. But, this does not ensure that the transition fires in time. Time for CPNs has been introduced as well. Van der

Aalst et al. [25] propose the use of interval timed colored Petri nets. Yet they are eager as well and transitions fire as soon as they are enabled. However, in planning, the progression of time may release resources allowing for an optimal non-greedy solution. For the modeling tool *CPN Tools*, Jensen et al. [26] introduce timestamps for tokens. Timestamps are attributes and can be edited by transitions describing similar semantics to our proposed approach.

6. Discussion and conclusion

In this paper, we presented an extension of fragment-based case management, introducing the resources involved in task executions and the duration of non-atomic tasks to case models. Both aspects were first elaborated and a syntactical extension was introduced before defining the formal semantics of the extensions by extending the existing mapping to CPNs.

The behavior of Petri nets and CPNs can be analyzed. In many cases, this requires the state space to be finite. For fCM without the proposed extensions, this is achievable by adhering to a number of assumptions [4]. To maintain a finite state space for the proposed extensions, these assumptions have to be extended. Firstly, we assume a finite set of resources to limit the number of possible task assignments. Secondly, the clock may not fire infinitely. Instead, we limit the number of consecutive clock firings to the maximum activity duration. This ensures that the behavior does not allow waiting infinitely long.

Despite being finite, a CPN's state space can still become large enough to make efficient analysis very challenging. A potentially large amount of available resource assignments per task leads to an increased state space complexity. At the same time, the necessity of resource availability for task enablement as well as data and resource locking and time constraints restrict possible execution paths, thereby reducing the state space. To that end, the impact of our extension on the state space complexity is a starting point for future investigation.

A limitation of the current syntax is that it cannot express tasks whose execution requires multiple resources of different roles. For now, we presented semantics similar to BPMN, where every task is assigned to a single lane. In addition, the defined semantics does not capture that the assignment of additional resources to a task may lead to a decreased task duration. To accomplish that, a duration function may be introduced in the future, mapping the number of resources to the expected duration.

The prototypical implementation presented in section 4 does not yet capture time ranges, but can be extended easily. Furthermore, the compiler of case models to CPNs [27] can be extended to automatically generate CPNs for the extended fCM models.

As a next step, the proposed extensions will be applied to other modeling approaches to evaluate their general applicability. Another interesting approach is the use of the extended CPNs' state space as input for planning algorithms to support knowledge workers in their decisions as previously presented [11]. For now, roles and resources are simply defined as sets of roles and resource instances with roles. In practice, organizational charts are frequently used. Their incorporation into fCM should be investigated.

In summary, we provide an extension of fragment-based case models to incorporate organizational models and time. This semantically defined extension lays the foundation for using case models to automatically compute planning support in knowledge-intensive processes.

References

[1] M. Weske, Business process management: concepts, languages, architectures, Springer, 2019.

[2] C. D. Ciccio, A. Marrella, A. Russo, Knowledge-intensive processes: Characteristics, requirements and analysis of contemporary approaches, J. Data Semant. 4 (2015) 29–57. doi:10.1007/s13740-014-0038-4.

[3] S. Steinau, A. Marrella, K. Andrews, F. Leotta, M. Mecella, M. Reichert, DALEC: a framework for the systematic evaluation of data-centric approaches to process management software, Softw. Syst. Model. 18 (2019) 2679–2716. doi:10.1007/s10270-018-0695-0.

[4] S. Haarmann, WICKR: A Joint Semantics for Flexible Processes and Data, Ph.D. thesis, Universität Potsdam, 2022.

[5] M. Hewelt, M. Weske, A hybrid approach for flexible case modeling and execution, in: M. L. Rosa, P. Loos, O. Pastor (Eds.), Business Process Management Forum - BPM Forum 2016, volume 260 of *Lecture Notes in Business Information Processing*, Springer, Cham, 2016, pp. 38–54. doi:10.1007/978-3-319-45468-9_3.

[6] P. Pyöriä, The concept of knowledge work revisited, J. Knowl. Manag. 9 (2005) 116–127. doi:10.1108/13673270510602818.

[7] S. K. Venero, J. C. dos Reis, L. Montecchi, C. M. F. Rubira, Towards a metamodel for supporting decisions in knowledge-intensive processes, in: C. Hung, G. A. Papadopoulos (Eds.), Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, SAC 2019, ACM, 2019, pp. 75–84. doi:10.1145/3297280.3297290.

[8] S. K. Venero, B. R. Schmerl, L. Montecchi, J. C. dos Reis, C. M. F. Rubira, Automated planning for supporting knowledge-intensive processes, in: S. Nurcan, I. Reinhartz-Berger, P. Soffer, J. Zdravkovic (Eds.), Enterprise, Business-Process and Information Systems Modeling - 21st International Conference, BPMDS 2020, 25th International Conference, EMMSAD 2020, volume 387 of *Lecture Notes in Business Information Processing*, Springer, 2020, pp. 101–116. doi:10.1007/978-3-030-49418-6_7.

[9] A. Marrella, Automated planning for business process management, J. Data Semant. 8 (2019) 79–98. doi:10.1007/s13740-018-0096-0.

[10] S. Haarmann, A. Seidel, M. Weske, Modeling Objectives of Knowledge Workers, in: A. Marrella, B. Weber (Eds.), Business Process Management Workshops, Lecture Notes in Business Information Processing, Springer International Publishing, Cham, 2022, pp. 337–348. doi:10.1007/978-3-030-94343-1_26.

[11] A. Seidel, S. Haarmann, M. Weske, Model-based decision support for knowledge-intensive processes, Journal of Intelligent Information Systems (2022). doi:10.1007/s10844-022-00770-0.

[12] K. Andree, L. Bein, M. König, C. Mandel, M. Rosenau, C. Terboven, D. Bano, S. Haarmann, M. Weske, Design-time support for fragment-based case management, in: C. Cabanillas, N. F. Garmann-Johnsen, A. Koschmider (Eds.), Business Process Management Workshops - BPM 2022 International Workshops, volume 460 of *Lecture Notes in Business Information Processing*, Springer, 2022, pp. 231–242. doi:10.1007/978-3-031-25383-6_17.

[13] T. Murata, Petri nets: Properties, analysis and applications, Proceedings of the IEEE 77 (1989) 541–580.

[14] R. M. Dijkman, M. Dumas, C. Ouyang, Semantics and analysis of business process models in BPMN, Inf. Softw. Technol. 50 (2008) 1281–1294. URL: https://doi.org/10.1016/j.infsof.2008.02.006. doi:10.1016/j.infsof.2008.02.006.

[15] Object Management Group, Business process model and notation (BPMN), 2014. URL: https://www.omg.org/spec/BMMN.

[16] K. Jensen, Coloured Petri Nets - Basic Concepts, Analysis Methods and Practical Use - Volume 3, Monographs in Theoretical Computer Science. An EATCS Series, Springer, 1997. URL: https://doi.org/10.1007/978-3-642-60794-3. doi:10.1007/978-3-642-60794-3.

[17] J. Eder, E. Panagos, M. Rabinovich, Time constraints in workflow systems, in: Advanced Information Systems Engineering: 11th International Conference, CAiSE" 99 Heidelberg, Germany, June 14—18, 1999 Proceedings 11, Springer, 1999, pp. 286–300.

[18] J. C. de A. R. Gonçalves, F. A. Baião, F. M. Santoro, G. Guizzardi, A cognitive BPM theory for knowledge-intensive processes, Bus. Process. Manag. J. 29 (2023) 465–488. doi:10.1108/BPMJ-11-2021-0746.

[19] Object Management Group, Case management model and notation (CMMN), 2016. URL: https://www.omg.org/spec/CMMN.

[20] M. Zur Muehlen, Resource modeling in workflow applications, in: Proceedings of the 1999 Workflow Management Conference, volume 70, Citeseer, 1999, pp. 137–153.

[21] M. Pesic, H. Schonenberg, W. M. P. van der Aalst, DECLARE: full support for loosely-structured processes, in: 11th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2007), IEEE Computer Society, 2007, pp. 287–300. URL: https://doi.org/10.1109/EDOC.2007.14. doi:10.1109/EDOC.2007.14.

[22] S. Remy, Incorporating organizational aspects into fragment-based case management, in: J. Manner, S. Haarmann, S. Kolb, O. Kopp (Eds.), Proceedings of the 12th ZEUS Workshop on Services and their Composition, Potsdam, Germany, February 20-21, 2020, volume 2575 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2020, pp. 10–17. URL: https://ceur-ws.org/Vol-2575/paper3.pdf.

[23] S. Cheikhrouhou, S. Kallel, N. Guermouche, M. Jmaiel, The temporal perspective in business process modeling: a survey and research challenges, Serv. Oriented Comput. Appl. 9 (2015) 75–85. URL: https://doi.org/10.1007/s11761-014-0170-x. doi:10.1007/s11761-014-0170-x.

[24] L. Popova-Zeugmann, L. Popova-Zeugmann, Time petri nets, Springer, 2013.

[25] W. M. van der Aalst, Interval timed coloured petri nets and their analysis, in: International conference on application and theory of petri nets, Springer, 1993, pp. 453–472.

[26] K. Jensen, L. M. Kristensen, L. Wells, Coloured petri nets and cpn tools for modelling and validation of concurrent systems, International Journal on Software Tools for Technology Transfer 9 (2007) 213–254.

[27] S. Haarmann, M. Montali, M. Weske, Refining case models using cardinality constraints, in: M. L. Rosa, S. W. Sadiq, E. Teniente (Eds.), Advanced Information Systems Engineering - 33rd International Conference, CAiSE 2021, Proceedings, volume 12751 of *Lecture Notes in Computer Science*, Springer, Cham, 2021, pp. 296–310. doi:10.1007/978-3-030-79382-1_18.