



# A Landscape for Case Models

Fernanda Gonzalez-Lopez<sup>1(✉)</sup> and Luise Pufahl<sup>2</sup>

<sup>1</sup> Pontificia Universidad Catolica Valparaiso, Valparaiso, Chile  
m.fernanda.gonzalez@gmail.com

<sup>2</sup> Hasso Plattner Institut, University of Potsdam, Potsdam, Germany  
luise.pufahl@hpi.de

**Abstract.** Case Management is a paradigm to support knowledge-intensive processes. The different approaches developed for modeling these types of processes tend to result in scattered models due to the low abstraction level at which the inherently complex processes are therein represented. Thus, readability and understandability is more challenging than that of traditional process models. By reviewing existing proposals in the field of process overviews and case models, this paper extends a case modeling language – the fragment-based Case Management (fCM) language – with the goal of modeling knowledge-intensive processes from a higher abstraction level – to generate a so-called fCM landscape. This proposal is empirically evaluated via an online experiment. Results indicate that interpreting an fCM landscape might be more effective and efficient than interpreting an informationally equivalent case model.

**Keywords:** Case Management · Process landscape · Process map · Process architecture · Process model

## 1 Introduction

Case Management (CM) is a paradigm to support the design, execution, monitoring, and evaluation of knowledge-intensive processes [20]. These types of processes are often found in domains where highly trained workers (i.e. *knowledge workers*) deal with very diverse units of work (i.e. *cases*). In fact, the term CM originated in the healthcare domain, where *medical personnel* – knowledge workers – deal with *patients* – cases – and the end-to-end process is not clear beforehand, but is rather tailored on-the-go based on aspects, such as examination results and medical team expertise.

In CM and analogous to a traditional *process model*, a *case model* represents all possible courses of action for handling cases in a given scenario. Different approaches have been developed for CM, most of them with a strong data-orientation. Business artifacts [23] and their Guard-Stage-Milestone (GSM) lifecycles [14] put data in the center of the approach. Based on GSM, the industry-standard CMMN (Case Management Modeling and Notation) [25] was designed. The *fragment-based Case Management* (fCM) [12] understands

knowledge-intensive processes as having structured parts – i.e. *process fragments* – that are flexibly combined at run-time based on data handled by the process. Regarding its notation, fCM reuses concepts from BPMN (Business Process Model and Notation) [24]; we call this the fCM-language. As the CM approaches capture complex behaviour of knowledge-intensive processes – including processed data artifacts, possible operations on them, and their interrelation – case models tend to include more concepts and are more scattered than traditional workflow-like process models. For capturing flexibility, the routing and the control flow might be more difficult to understand compared to an imperative sequence flow [31], such that Lantow [16] reports a lack of understandability of CMMN models.

Several works have been developed to provide accessibility to a comprehensive functional description of a business [17]. In such level of abstraction, individual processes are depicted as black boxes and, therefore, the focus of the model is on the structure of the collection of processes [8]. By analogy, this view could be used to depict process fragments within a case model. In this paper, we use the term *process overviews* to refer either to *process maps* [17, 18], *process landscapes* [3, 10], or *process architectures* [6, 9]. In the range of possibilities of process overviews, process landscapes stand as the middle ground between the less-technical process maps and the more-technical process architectures [8]. Compared to detailed process models, process overviews allow to represent in a more straightforward way: (a) high-level concepts regarding to a single process, such as inputs/outputs; as well as (b) concepts regarding the relationships between processes, such as trigger and data flow. These concepts are either not available or indirectly represented in current approaches for case modeling.

This paper extends the fCM-language for modeling overviews of knowledge-intensive processes. The goal is making case models more accessible and understandable, and thus easier to analyze by their users. We classify these models as *case model landscapes* (CMLs) since we expect the proposal to be, on one side understandable by non-technical users, but also useful for technical ones. We focus on the fCM approach [12]; still we will discuss its application to other approaches. Existing languages for modeling cases and process overviews are reviewed and their usefulness for CMLs is discussed. Based on the found limitations, we develop a language<sup>1</sup> for CMLs as extension of the fCM-language. The proposal is evaluated in comparison to the non-extended fCM-language in an online experiment where the participants are asked to answer questions on two business scenarios represented in these two languages. The correctness of the answers as well as the time needed is measured to assess *interpretation efficiency* and *effectiveness* as proposed by [4, 18].

In the remainder, related work on case management and process overviews is discussed in Sect. 2. Then, requirements for a CML and different alternatives are

---

<sup>1</sup> A *language* is a structured set of symbols whose combination represents concepts which carry a certain meaning. A language is specified using a meta-model describing its *abstract syntax* (i.e. constituting concepts and their relations) and its *semantics* (i.e. meaning of the concepts).

presented in Sect. 3. The extension of the fCM-language for CML is presented in Sect. 4 and its empirical evaluation is discussed in Sect. 5, followed by conclusions in Sect. 6.

## 2 Related Work

In this section, related work regarding case management and attempts to ease the case model understanding, and approaches for process overviews are presented.

*Case Management.* A first approach for capturing case models has been introduced as *Case Handling* in [1,2], which led to shifting the focus from activities to data. *Business Artifacts* [23] with the Guard-Stage-Milestone (GSM) approach [14] focus on the high-level data artifacts handled during case processing. This was used as the basis for the CMMN (Case Management Modeling and Notation) [25] standard which allows to specify, for example, optional and non-optional parts of a case and milestones that need to be reached. However, some aspects of data – essential for case management (CM) – cannot be represented using CMMN. Despite an existing standard, other CM approaches were still continued or newly developed, most prominently PHILharmonicFlows [15], fragment-based Case Management (fCM) [12], and the declarative approach [27]. PHILharmonicFlows [15] splits a process into micro processes describing how a data artifact can be changed and macro processes handling micro processes relations. To deal with complexity, Steinau et al. [29] propose relational process structures representing the relationships between processes with cardinalities. However, aspects, such as the results exchanged by the process fragments or the trigger relations are not captured, limiting the understanding and the analysis of such a case model. fCM by Hewelt and Weske [12] combines process fragments at runtime according to data conditions. In [11], Hewelt et al. provide a method for supporting the case model elicitation. Still, it is an open challenge that the resulting case model is difficult to read for people not involved in the case model design. The declarative approach [27] tries to avoid the disadvantages of imperative process models and allows more flexibility by defining constraints and rules between activities, whereby produced and consumed data of the activities is not considered. However, experiments showed that declarative process models seem to be more difficult to comprehend [28]. Therefore, De Smed et al. [5] propose dependencies diagrams to visualize implicit dependencies between actions in declarative models. It has a quite low abstraction level which might lead to understandability issues in case of more complex models. Furthermore, it builds upon on the constraint concepts of declarative models with no graphical elements targeting more declarative modeling experts.

*Process Overviews.* Process overviews – a term used in this paper for referring either to a process map, landscape, or architecture – support reasoning and analysis of the structure of the process collection, leaving aside much detail of individual processes [8]. Commonly, process overviews address the concerns

of business-oriented users but they can also address the concerns of technical-oriented ones. Consequently, a language to express such a model aims to be easily understood by a non-technical audience [9]. Process maps are usually easily readable by non-technical users due to being modeled with a small set of concepts with a lax semantics. It might consist solely of a hierarchical classification of processes, or also that inputs/outputs of the constituting processes are specified [19]. Process architectures are more technically oriented and each represented concept has a precise semantics. For example, the approach by Eid-Sabbagh et al. [6] provides information, about trigger and resource flow relationships between processes based on events. In the extension of this work, exclusive, sequential, and interaction relations between processes are discovered based on the data they handle, however data is not explicitly represented in the architecture model [7]. Process landscapes could be seen as the middle ground between process maps and architectures. Proposals in this area also struggle with the issue of ensuring an adequate level of understandability, e.g. [3, 10]. Altogether, multiple approaches have been proposed to convey overviews for collections of processes. We argue that the therein used concepts could be adapted for building overviews for case models. In order to do so, it would be necessary to abstract from the details of process fragments and rather focus on the way they relate to each other. This is similar to the dependencies diagrams proposed by De Smed et al. [5]. However, our proposal places the emphasis on data-based relationships and a intuitively understandable graphical language.

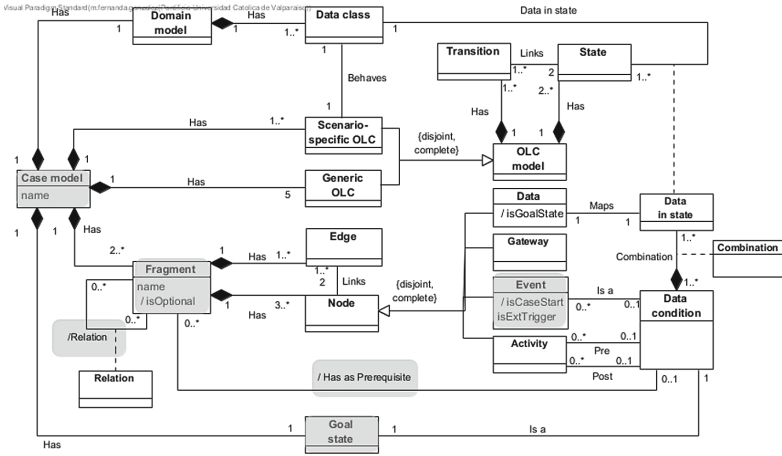
### 3 Requirements for a Case Model Landscape

Section 3.1 introduces the fCM-language using the meta-model in Fig. 1 and the health-care example in Fig. 2. Then, requirements for a fCM landscape are defined in Sect. 3.2. Finally, alternative landscape approaches and their limitations are discussed in Sect. 3.3. In the remainder of the paper, we use a *medical consultation* business scenario to illustrate the discussed concepts. In the example, when a patient arrives to the hospital, she will be attended by a medical team for providing diagnosis and treatment and also by personnel for administrative matters, all with the goal of sending her healthy back home.

#### 3.1 Fragment-Based Case Management Language

Figure 1 shows the meta-model that specifies the fCM-language, based on the specifications in [12]. In fCM, a case model consists of four artifacts to be detailed in the following: (a) a *domain model*, (b) a set of *object lifecycles*, (c) a *goal state*, and (d) a set of *process fragments*.

*Domain Model.* The domain model represents the static view of the data that is relevant to the scenario. As portrayed in Fig. 1 (upper section), it is composed of a collection of data classes defining relevant data types and their data attributes. In the example in Fig. 2a, the relevant data types are **Biopsy**, **Patient File**, **X-ray**, and **Tomography**.



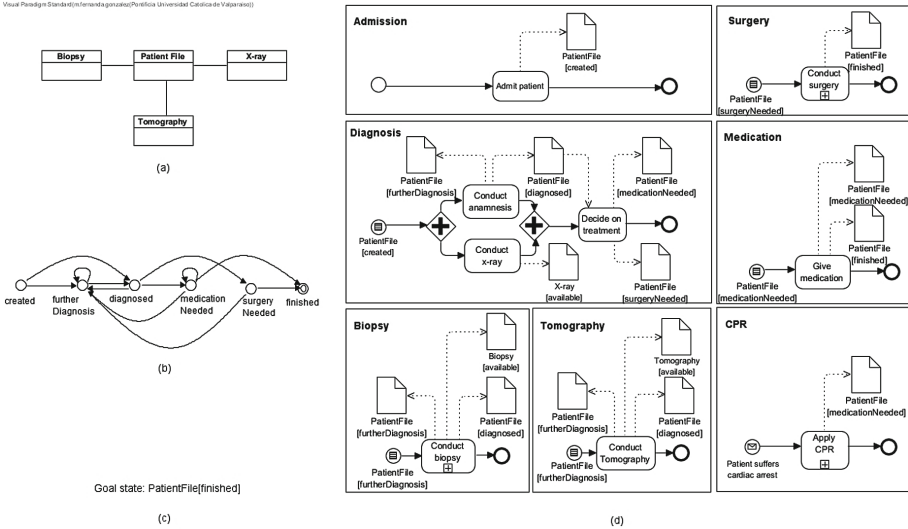
**Fig. 1.** Meta-model for fCM-language (fCML requirements highlighted)

*Object Lifecycles.* As showed in Fig. 1 (middle upper section), every class of the domain model behaves according to a scenario-specific object lifecycle (OLC). An OLC depicts possible states and transitions that an instance of a certain data type may undergo during the handling of a case. Figure 2b shows the OLC of the *Patient file* as a finite state machine with the following possible states: *created*, *furtherDiagnosis*, *diagnosed*, *medicationNeeded*, *surgeryNeeded*, and *finished*. Additionally, a set of *generic OLCs* is pre-defined in fCM for the execution semantics of cases, fragments, activities, gateways, and events.

*Goal State.* The goal state defines when a case model instance may terminate in terms of a logical combination of a subset of all possible classes in their OLC-defined states, as showed in Fig. 1 (bottom section). Figure 2c shows the goal state for our running example: a *Patient File* in state *finished*.

*Process Fragments.* A case model contains multiple process fragments as showed in Fig. 1 (middle lower section). In the example, the fragments are *Admission*, *Diagnosis*, *Surgery*, *Medication*, *Biopsy*, *Tomography*, and *Cardiorespiratory Resuscitation (CPR)*, as depicted in Fig. 2d. As showed in the meta-model, each process fragment is composed by a set of data, gateway, event, and activity nodes linked by flow edges, as in traditional process models. Fragment modeling requires consistency in labeling data objects to capture the relations between fragments. In Fig. 2d data types and their states are depicted using BPMN data-object notation: *Object type [state]*. As in many CM approaches, data is the key element around which fCM process fragments are organized. Figure 1 (right section) shows that data conditions are defined as the combination of data class type in some state of their OLCs. On one hand, a start event of a fragment could be itself a data condition, which means that such fragment is only enabled to start once a given data condition is true. For example, and as showed in

Fig. 2d, the *Diagnosis* fragment becomes enabled when there is a data instance of **Patient file** [created]. On the other hand, activities within the different fragments read/write data in a given state. For example, in Fig. 2d the *Admission* fragment writes a **Patient file** [created]. It is possible to identify a relation between the *Admission* and the *Diagnosis* fragments, since the output of the former, enables the execution of the latter.



**Fig. 2.** Partial fCM case model for medical consultation: (a) domain model, (b) object lifecycle for the **Patient File**, (c) goal state, and (d) process fragments.

### 3.2 Requirements

As showed in our running example (see Fig. 2), the information within an fCM case model is scattered between various sub-models. In our experience (e.g. [11]), this poses a challenge for the readers using the model to answer simple high-level questions, such as *Where does the knowledge-intensive process start?* This issue can be tackled by creating a more abstract view, where some information from the case model is hidden and some information is made more straightforward/accessible. For fCM, we name such a view an fCM-landscape (fCML). We define requirements for fCML based on: (i) particularities of CM and knowledge-intensive processes as modeled using fCM, (ii) research on Process Overviews, and (iii) available standards in the fields of CM and Process Overviews. Requirements are described in the following and are showed as highlighted elements in the fCM-language meta-model in Fig. 1:

- **Business scenario.** Approaches for CM and Process Overviews consider often a – sometimes implicit – container specifying the limits of what lies

- within the business scenario (e.g. [25]) or process collection (e.g. [18]), respectively. This concept is also important for fCML as it defines a case model as a container for a set of fragments and data objects to reach a certain goal. This is showed in the *name* attribute of the `Case model` class in Fig. 1.
- **Case start.** In fCM, the start of a knowledge-intensive process is represented as a BPMN blank start event in the first fragment that can be executed. Neither in CMMN nor in Process Overviews is this distinction required, though it might be represented explicitly as an event listener [25], or implicitly by the sequence of processes [6] or the input for a process map [18]. However, we define it as a requirement for fCML due to being relevant for fCM. This is expressed by the derived attribute *isCaseStart* of the `Event` class in Fig. 1.
  - **Goal state.** Another key feature of fCM to be included in the fCML, is the definition of a data condition for ending the case, represented by the `Goal state` class in Fig. 1. This concept relates to process map outputs [18].
  - **Fragment.** The central concept of a Process Overview is the process depicted as a labeled black box [6,19]. By analogy, the process fragment should be defined as the central concept of a fCML. This requirement is showed in the *name* attribute of the `Fragment` class in Fig. 1.
  - **External trigger.** Process Overviews consider that processes might be triggered by events [6]. Analogously, we then define that a fCML should provide information regarding triggering of fragments via external events. This is showed in the *isExtTrigger* attribute of the `Event` class in Fig. 1.
  - **Pre-requisite.** The fact that some fCM fragments need to be data-enabled to be executed is similar to the concept of processes needing an input in a Process Overview (e.g. [18]). This fCML requirement is considered in the *Has as Prerequisite* derived association in Fig. 1.
  - **Fragment relations.** *Data-flow relations* between processes are data-related aspects usually visualized in Process Overviews [19]. A key aspect of fCM is that the relations between fragments are based on data. Therefore, this concept is considered as an fCML requirement as showed on the *Relation* derived association in Fig. 1. Concepts like exclusiveness, sequential dependency, and interaction proposed by Eid-Sabbagh et al. [7] are of high relevance.
  - **Fragment optionality.** A central aspect of CM are process fragments combination depending on the case at hand. Accordingly, CMMN defines that some parts of the case model can be discretionary. We rank this concept as important for end users to highlight the optional fragments which do not need to be executed for all possible cases. This fCML requirement is showed as the *isOptional* derived attribute of class `Fragment` in Fig. 1.

### 3.3 Alternatives

Together with fCM [12], a set of languages for Process Overviews and CM approaches was assessed to find out whether they provided the means to fulfill the requirements for an fCML previously discussed in Sect. 3. The justification for selecting these works, is that they are either the industry standards in their fields – ArchiMate [30] and CMMN [25] –, or they are representative and well

documented proposals from the research community – Process Architecture by Eid-Sabbagh [6, 7] and Process Maps by Malinova [18]. A summary of the results is presented in Table 1 and discussed in detail in the following.

**Table 1.** Alternatives, where ✓: full support, -: partial support, and X: no support.

	ArchiMate [30]	CMMN [25]	fCM [12]	Process architecture [6, 7]	Process map [18]
Business scenario	-	✓	-	-	-
Case start	-	✓	✓	-	-
Goal state	-	X	✓	X	-
Fragment	-	✓	✓	-	-
External trigger	-	✓	✓	-	-
Pre-requisite	X	-	✓	-	-
Fragment relations	-	-	-	✓	-
Fragment optionality	-	✓	-	-	X

*ArchiMate.* ArchiMate is an architecture description language for enabling unambiguous description, analysis, and visualization of the relationships among business domains [30]. This language has become an industry standard for modeling enterprise architectures, and therefore, can be used to model Process Architectures. As showed in Table 1, ArchiMate supports most of the requirements for fCML, but only in a partial way due to being a general purpose language.

*CMMN.* As a modeling standard for CM, CMMN [25] fulfills many of the fCML requirements, as showed in Table 1. The weak points of CMMN are, however, those related to data, namely goal state, data pre-requisites, and data-aspects of fragments relations. An interesting aspect of CMMN is the concept of *sentries* (cf. [14]), which stand for entry and exit conditions of fragments.

*fCM.* The fCM approach [12] has been already described in detail in previous sections. As showed in Table 1, fCM supports the fCML requirements either fully or partially. In line with what we have previously discussed, the limitation of fCM is the scattered information among its various models.

*Process Architecture.* The approach to Process Architecture by Eid-Sabbagh [6, 7] provides a language for describing process architectures. This language, however, does not consider goal states, as showed in Table 1. Data considerations are rather implicit in the architecture model: they provide a conceptual ground for defining some inter-process relations in [7]. Two particularities of this approach are the strong focus on events and the fact that it defines exclusiveness, sequential dependency, and interaction between processes.

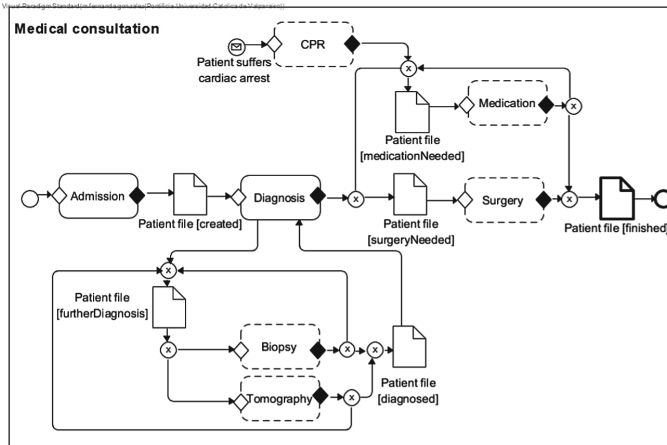


*Process Map.* The Process Map approach by Malinova [18] provides partial support of most of the fCML requirements, as showed in Table 1. The main limitation of this language for modeling fCML is, again, related to data. Being a business-oriented model, data-flow between processes is considered at a very high level of abstraction, leaving outside details regarding data handling. The language provides the concept of condition, which semantic is not described in detail, but that somehow relates to the CMMN notion of sentries.

Results of the analysis proved none of the approaches was entirely suitable for the task of modeling a fCML. However, they ground our proposal (see Sect. 4).

## 4 Extension of fCM-Language for Modeling Landscapes














After identifying its requirements, this section introduces the extension of the fCM-language for modeling a fCML. We decided to re-use notational elements from BPMN and CMMN – both standards of the Object Management Group – due to having a high recognition factor by business people working with process models. We mostly reuse the notational elements of BPMN and CMMN in such a way that they still have the original meaning. The elements of the proposal, their semantic meaning, and the notation is given in Table 2. We will introduce the language extension based on the running example of the *medical consultation*: Fig. 3 shows the equivalent fCML for the case model in Fig. 2.



**Fig. 3.** Case Model Landscape for the medical consultation scenario.

The case model always starts with the *Admission* fragment, which follows the blank start event. Each fragment can have a pre-requisite and an output which are shown as unfilled and filled diamonds at the borders of the fragments similar to the entry and exit criteria in CMMN. The pre-requisite describes the condition

**Table 2.** Modeling elements of the fCM-language extension for fCML.

Element	Description	Notation
Business scenario	Container of a landscape for a specific case model.	
Blank start event	Start of a case model. If a new case of the model is instantiated then it is started with the succeeding process fragment.	
Message start event	External occurrence of an event, which is relevant for the case. It enables the start of the succeeding process fragment.	
Process fragment, non-optional	Non-optional process fragment that needs to be executed in every possible execution of a case model.	
Process fragment, optional	Optional process fragment that is not necessarily executed in every possible execution of a case model.	
Pre-requisite	Data pre-condition or an event that enables the start of a process fragment.	
Output	Output produced by a process fragment, in terms of data. It is optional to include it if it is not required by another process fragment.	
Connector	Causal relation between the elements of the case model.	
Logic operator, AND	Forking or merging of paths following the logic of a logical AND-operator.	
Logic operator, OR	Forking or merging of paths following the logic of a logical OR-operator.	
Data object	Data type holding a particular state in which it is available as an input or an output of a process fragment.	
Data object, goal state	Data condition that must be fulfilled for the case to terminate. This can also be the combination of data conditions via logic operators.	
End event	End of a case model, it is enabled due to achieving the termination condition of the case model.	

that must be satisfied to start a fragment and the output describes the data outcomes produced by a fragment. The *Admission* fragment has no data input condition – it simply starts by initiating a new case – but it produces as output the `PatientFile[created]`, needed as pre-requisite by the *Diagnose* fragment.

During the execution of the *Diagnose* fragment, a `PatientFile[furtherDiagnosis]` can be produced which is visualized by an outgoing arc from the fragment connected to the data object. If the data object is available, the optional *Biopsy* fragment or the optional *Tomography* fragment, or both can be executed. This construct is represented by a logical OR-operator connected to the prerequisite of both fragments. These two fragments do not need to be executed in every case, they are optional which is shown by a dotted boarder line similar to the discretionary tasks/states in CMMN. The output of both fragments can be the `PatientFile` in *furtherDiagnosis* or *diagnosed* also represented with the help of a logical OR operator. In case of *furtherDiagnosis*, the two just discussed fragments can be restarted. In the other case, the *Diagnose* fragment is continued, which is shown by the incoming connector into the fragment box.

This fragment produces as output either the `PatientFile` in *surgeryNeeded* or *medicationNeeded* triggering the optional fragments *Surgery* or *Medication*, respectively. Both the fragments can produce `PatientFile[finished]` representing the goal state of the case model and leading to the end event, the end of the case model. The *Medication* fragment can also result in `PatientFile[medicationNeeded]` as alternative, re-triggering this fragment.

During the case execution, also a relevant event for this business scenario can occur – *Patient suffers from cardiac arrest*. Represented by a message start event, this event triggers the *CPR* fragment. It also results in the `PatientFile[medicationNeeded]` object. The logical OR connector above this data object implies that `PatientFile[medicationNeeded]` can be result of three fragments: the *Diagnose*, the *Medication*, or the *CPR* fragment. Here, the AND connector was not applied. This can be used to represent the need of several data objects to trigger a fragment, or different data objects are produced as output.

## 5 Evaluation

An experiment was design to assess our proposal. The experimental design is described in Sect. 5.1, and results are presented and discussed in Sect. 5.2.

### 5.1 Experimental Design

The independent variable of the experiment is the case modeling language: the proposed extension vs. the fCM-language (as discusses in Sect. 3.2, no other analyzed approach supports all requirements). Following [4], the experiment dependent variables are *interpretation effectiveness* – i.e. how faithfully does the interpretation of the model represents the semantics of the model –, *interpretation effort* – i.e. amount of resources needed to interpret the model –, and *interpretation efficiency* – quotient of them both. In this regard, the hypotheses we aimed to test were whether interpretation of case models is less effective ( $H1_0$ ), requires more effort ( $H2_0$ ), and is less efficient ( $H3_0$ ) when using the fCM-language than when using the proposed extension. For testing these three hypotheses, we used paired Wilcoxon signed rank test, the non-parametric version of the paired t-test (see [13]). The grounds for using non-parametric statistics for data analysis is that, as showed in Fig. 4, no assumption of normality could be made about the collected data. Statistical analysis in this study considered a 95% confidence.

The subjects were students from the Hasso Plattner Institute, University of Potsdam, who were invited to voluntarily join the experiment. These students are easy accessible representatives of the target audience of case models. To maximize data collection, the experiment followed a crossover design in which each subject read a case model of one business scenario in fCM-language (control treatment or C) followed/preceded by reading a case model of another business scenario in the proposed extension (experimental treatment or E). The business scenarios used were *traumatology emergency* [22] (H) and *organization of a business trip* [11] (B), and their control and treatment model variants were

designed to be informationally equivalent and were available during the whole experiment as recommended by Parson and Cole [26]. Altogether, this resulted in the following four treatments: EH/CB (treatment A), CB/EH (treatment B), EB/CH (treatment C), and CH/EB (treatment D). For example, treatment A corresponds to exposure to, firstly, the experimental treatment using the traumatology emergency scenario and, secondly, to the control treatment using the business trip organization scenario. We used block random assignment of the subjects according to the initial letter of their last name.

The experiment was conducted online using *Google Forms*<sup>2</sup>. We first defined a set of design-time and run-time aspects of case models (e.g. *case start*, *fragment repetition*), and then a set of 20 true or false statements addressing those aspects. For example, to address the *case start* aspect we formulated the following question: *In all cases, fragment X is the first to be executed*, where *X* is the name of a fragment in a given case model. For each respondent, she was firstly asked demographic questions. Before reading each model, the respective language was explained to her, and afterwards she was asked to answer the set of questions regarding a model of one of the business scenarios. Then this was repeated for the other business scenario using the other language. Interpretation effectiveness was measured as the total score of the set of questions (1 point per correct answer), interpretation effort was measured as the total time (in minutes) she used to complete the task, and interpretation efficiency was measured as the quotient of the previous variables.

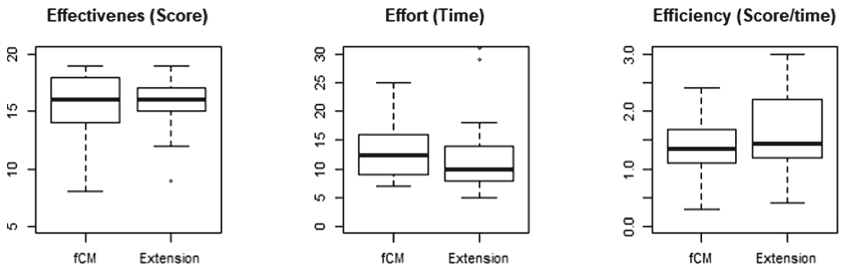
## 5.2 Results and Discussion

The 24 subjects of the study were classified as novice or experienced, according to the modeling courses they had undertaken: one or more. Compared to the experienced subjects (8 in total), the novice subjects (16 in total) self-reported lower BPMN and CM experience but higher domain knowledge on the traumatology emergency and business trip scenarios. The overall low self-reported domain knowledge is desirable since it prevents subjects from answering questions based on prior domain knowledge rather than on model interpretation [4].

Figure 4 summarizes data gathered in the experiment after discarding two problematic observations. Overall and leaving outside out-layer observations, data in Fig. 4 for interpretation effectiveness and effort is shifted towards better performance for our proposal. Regarding average interpretation effectiveness, its value was slightly higher for the extension (15.8/20 points) than for the fCM-language (15.5/20 points). Again in average, interpretation effort led to slightly better results when using the proposal (12 min) in comparison to the fCM-language (13.3 min). Average interpretation efficiency, consequently, follows the same pattern of the proposal (1.6 points/min) slightly outperforming the fCM-language (1.3 points/min). As showed in Fig. 4, it is also possible to observe a higher dispersion of both scores and time is found for the fCM-language.

<sup>2</sup> Forms and raw data available at: <https://drive.google.com/drive/folders/1c-ZZ6HA6H7d7yOgthcoVANLt-wnRfOhS?usp=sharing>.

This might indicate a desirable feature of the proposal: leading to more consistent interpretation of case models in terms of effectiveness and effort.



**Fig. 4.** Descriptive statistics of the experimental dependent variables.

Hypothesis testing provided no significant evidence to reject  $H_{10}$  ( $p=0.2605$ ),  $H_{20}$  ( $p=0.9327$ ), nor  $H_{30}$  ( $p=0.7537$ ). This means that the data in our experiment weakly supports the thesis that the proposed extension outperforms the fCM-language in effectiveness, effort, or efficiency. We conducted additional tests to verify aspects that might have influenced the results using the Spearman rank-order correlation test [13]. By this, we were able to rule out the influence of treatment order (first C or E), lecture-based and self-reported BPMN/CM modeling experience, and self-reported prior domain knowledge. A limitation of our work is that we ensured similar complexity between the models used for the experiment – measured as the number of nodes [21] – based only on control models. However, the experimental versions of the models did not have a similar number of nodes due to fragment inter-relations leading to having treatments with different difficulty levels. A Spearman correlation test then indicated a significant direct relation between treatment difficulty – valued as 0 for treatments A and B, and as 1 to treatments C and D – and interpretation effort ( $p=0.0041$ ). We believe that this is an issue that might have negatively impacted our results and that, avoiding it, might lead to improving significance of the evidence supporting the benefits of our proposal. An additional aspect that might contribute to improving our results in future versions of the experiment would be to conduct it in a laboratory setting such that, for example, time measures are more accurate.

## 6 Conclusions

This paper provided a new concept for case management by presenting a means for modeling case model landscapes. This contribution is built upon the creation of a meta-model for extending the fragment-based Case Management (fCM) language. A case model landscape (CML) gives end users an integrated, comprehensive overview of the high-level activities and the processed data during the

execution of a knowledge-intensive process instead of the detailed case models with often scattered information about actions and data in different models. It can be used to get an understanding, but also to analyze case models, redesign, or check compliance requirements. As the landscapes builds up on the fCM approach, we tested its interpretation performance in an online experiment with students. The experiment results implicate that the proposal might improve interpretation of high-level aspects of case models, and that it may lead to more consistent interpretation of the models in terms of effectiveness and effort. These results should be, nonetheless, validated with further experimentation and consider a laboratory environment for having more reliable time data.

The proposed fCM-language extension for CML re-uses notational elements of the two modeling standards, BPMN (Business Process Modeling and Notation) and CMMN (Case Management Modeling and Notation), having the advantage that it might be easier understandable by business people working with process models. Still, it has the risk of some minor miss-interpretation which need to be further tested. The proposal could be also used for CMMN models, whereby stages and their relation could be shown on an abstract level. CMMN represents data mainly implicitly, our language represents data and data relations explicitly. Furthermore, the approach might be also interesting for PHILharmonicFlows, another relevant case management approach, to represent the relation between the micro processes. An important concept for PHILharmonicFlows are the cardinalities between the generated objects. These are only implicitly given in the proposed landscape by distinguishing between optional and mandatory fragments, and the possibility to trigger certain fragments more than once. An explicit representation might be a useful extension. In this work, so far the language for CML was presented, but not how to design or automatically derive it. On this, we want to focus in our future research.

## References

1. van der Aalst, W., Berens, P.: Beyond workflow management: product-driven case handling. In: 2001 International ACM SIGGROUP Conference on Supporting Group Work, pp. 42–51. ACM (2001)
2. van der Aalst, W., Weske, M., Grünbauer, D.: Case handling: a new paradigm for business process support. *Data Knowl. Eng.* **53**(2), 129–162 (2005)
3. Becker, J., Pfeiffer, D., Räckers, M., Fuchs, P.: Business Process Management in Public Administrations - The PICTURE Approach. In: PACIS 2007, Auckland, New Zeland, July 3–6, pp. 1–14 (2007)
4. Burton-Jones, A., Wand, Y., Weber, R.: Guidelines for empirical evaluations of conceptual modeling grammars. *J. Assoc. Inf. Syst.* **10**(6), 495–532 (2009)
5. De Smedt, J., De Weerd, J., Serral, E., Vanthienen, J.: Discovering hidden dependencies in constraint-based declarative process models for improving understandability. *Inf. Syst.* **74**, 40–52 (2018)
6. Eid-Sabbagh, R.-H., Dijkman, R., Weske, M.: Business process architecture: use and correctness. In: Barros, A., Gal, A., Kindler, E. (eds.) BPM 2012. LNCS, vol. 7481, pp. 65–81. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-32885-5\\_5](https://doi.org/10.1007/978-3-642-32885-5_5)

7. Eid-Sabbagh, R.-H., Hewelt, M., Meyer, A., Weske, M.: Deriving business process data architectures from process model collections. In: Basu, S., Pautasso, C., Zhang, L., Fu, X. (eds.) ICSSOC 2013. LNCS, vol. 8274, pp. 533–540. Springer, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-45005-1\\_43](https://doi.org/10.1007/978-3-642-45005-1_43)
8. Gonzalez-Lopez, F., Bustos, G.: Business process architecture design methodologies - a literature review. *Bus. Process Manag. J.* (2019). <https://doi.org/10.1108/BPMJ-09-2017-0258>
9. Green, S., Ould, M.: The primacy of process architecture. In: CAiSE Workshops (2), pp. 154–159 (2004)
10. Gruhn, V., Wellen, U.: Analysing a process landscape by simulation. *J. Syst. Software* **59**(3), 333–342 (2001)
11. Hewelt, M., Wolff, F., Mandal, S., Pufahl, L., Weske, M.: Towards a methodology for case model elicitation. In: Gulden, J., Reinhartz-Berger, I., Schmidt, R., Guerreiro, S., Guédria, W., Bera, P. (eds.) BPMDS/EMMSAD -2018. LNBIP, vol. 318, pp. 181–195. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-91704-7\\_12](https://doi.org/10.1007/978-3-319-91704-7_12)
12. Hewelt, M., Weske, M.: A hybrid approach for flexible case modeling and execution. In: La Rosa, M., Loos, P., Pastor, O. (eds.) BPM 2016. LNBIP, vol. 260, pp. 38–54. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-45468-9\\_3](https://doi.org/10.1007/978-3-319-45468-9_3)
13. Hollander, M., Wolfe, D.A., Chicken, E.: *Nonparametric Statistical Methods*, 3rd edn. Wiley, Hoboken (2014)
14. Hull, R., et al.: Business artifacts with guard-stage-milestone lifecycles: managing artifact interactions with conditions and events. In: DEBS 2011, pp. 51–62. ACM (2011)
15. Künzle, V., Reichert, M.: PHILharmonicFlows: towards a framework for object-aware process management. *J. Software Maintenance Evol. Res. Pract.* **23**, 205–244 (2011)
16. Lantow, B.: Adaptive case management - a review of method support. In: Buchmann, R.A., Karagiannis, D., Kirikova, M. (eds.) PoEM 2018. LNBIP, vol. 335, pp. 157–171. Springer, Cham (2018). [https://doi.org/10.1007/978-3-030-02302-7\\_10](https://doi.org/10.1007/978-3-030-02302-7_10)
17. Lunn, K., Sixsmith, A., Lindsay, A., Vaarama, M.: Traceability in requirements through process modelling, applied to social care applications. *Inf. Software Technol.* **45**(15), 1045–1052 (2003)
18. Malinova, M.: *A Language for Designing Process Maps*. Ph.D. thesis, Vienna University of Economics and Business (2016)
19. Malinova, M., Leopold, H., Mendling, J.: An explorative study for process map design. In: Nurcan, S., Pimenidis, E. (eds.) CAiSE Forum 2014. LNBIP, vol. 204, pp. 36–51. Springer, Cham (2015). [https://doi.org/10.1007/978-3-319-19270-3\\_3](https://doi.org/10.1007/978-3-319-19270-3_3)
20. Marin, M.A., Hauder, M., Matthes, F.: Case management: an evaluation of existing approaches for knowledge-intensive processes. In: Reichert, M., Reijers, H.A. (eds.) BPM 2015. LNBIP, vol. 256, pp. 5–16. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-42887-1\\_1](https://doi.org/10.1007/978-3-319-42887-1_1)
21. Mendling, J., Reijers, H.A., Cardoso, J.: What makes process models understandable? In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 48–63. Springer, Heidelberg (2007). [https://doi.org/10.1007/978-3-540-75183-0\\_4](https://doi.org/10.1007/978-3-540-75183-0_4)
22. Mertens, S., Gailly, F., Poels, G.: Enhancing declarative process models with DMN decision logic. In: Gaaloul, K., Schmidt, R., Nurcan, S., Guerreiro, S., Ma, Q. (eds.) CAiSE 2015. LNBIP, vol. 214, pp. 151–165. Springer, Cham (2015). [https://doi.org/10.1007/978-3-319-19237-6\\_10](https://doi.org/10.1007/978-3-319-19237-6_10)

23. Nigam, A., Caswell, N.S.: Business artifacts: an approach to operational specification. *IBM Syst. J.* **42**(3), 428–445 (2003)
24. OMG: Business Process Model and Notation (BPMN), V. 2.0 (2011)
25. OMG: Case Management Model and Notation (CMMN) V. 1.1 (2016)
26. Parsons, J., Cole, L.: What do the pictures mean? guidelines for experimental evaluation of representation fidelity in diagrammatical conceptual modeling techniques. *Data Knowl. Eng.* **55**(3), 327–342 (2005)
27. Pesic, M., van der Aalst, W.M.P.: A declarative approach for flexible business processes management. In: Eder, J., Dustdar, S. (eds.) *BPM 2006*. LNCS, vol. 4103, pp. 169–180. Springer, Heidelberg (2006). [https://doi.org/10.1007/11837862\\_18](https://doi.org/10.1007/11837862_18)
28. Pichler, P., Weber, B., Zugal, S., Pinggera, J., Mendling, J., Reijers, H.A.: Imperative versus declarative process modeling languages: an empirical investigation. In: Daniel, F., Barkaoui, K., Dustdar, S. (eds.) *BPM 2011*. LNBIP, vol. 99, pp. 383–394. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-28108-2\\_37](https://doi.org/10.1007/978-3-642-28108-2_37)
29. Steinau, S., Andrews, K., Reichert, M.: The relational process structure. In: Krogstie, J., Reijers, H.A. (eds.) *CAiSE 2018*. LNCS, vol. 10816, pp. 53–67. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-91563-0\\_4](https://doi.org/10.1007/978-3-319-91563-0_4)
30. The Open Group: *ArchiMate 3.0.1 Specification* (2017)
31. Zensen, A., Küster, J.: A comparison of flexible BPMN and CMMN in practice. In: *EDOC 2018*, pp. 105–114. IEEE (2018)