Fragment-Based Case Management Models: Metamodel, Consistency, and Correctness

Stephan Haarmann

Hasso Plattner Institute, University of Potsdam, Potsdam, Germany
stephan.haarmann@hpi.de

Abstract. Knowledge-intensive processes are inherently complex. Thus, modeling them is hard as the models have to capture various perspectives often using sub-models with hidden dependencies, e.g., the behavior of a process is constrained by cardinality constraints in a domain model. Yet, models are desirable as they are the primary tool in business process management to analyze, design, implement, and enact processes. We present a metamodel, consistency and correctness criteria for fragment-based case management. The criteria can i) be verified automatically and ii) used to assist modelers at design-time. Thereby, mistakes can be detected and even prevented during design, so model quality improves.

Keywords: Business Process Management, Process Modeling, Case Management

1 Introduction

Managing knowledge-intensive business processes is hard, and traditional business process management (BPM) is insufficient for this task [16]. While traditional business processes are highly structured, well-defined, and repetitive, knowledge-intensive ones are emergent, multi-variant, data-driven, and non-repeatable [3]. While highly structured processes are often modeled using imperative modeling languages, purely imperative models of knowledge-intensive processes are often perplexing.

Knowledge-intensive processes are executed by domain experts called knowlegde-workers, such as physicians, lawyers, and insurance clerks. The curse of the process is primarily determined by the decisions of the knowledge-workers, which are based on experience and case-specific information. Furthermore, knowledge-intensive processes are usually human-centered with little to no automation.

Novel modeling approaches fitted to knowledge-intensive processes have been proposed. Among them are case management approaches [1, 17]. In case management, knowledge workers gather, create, and maintain data. A case (process instance) evolves around this data. Every case has one central object, the case object (sometimes case folder). A case model describes both the data and the activities involved in a case as well as dependencies among them. Case management approaches can model knowledge-intensive processes more concisely than imperative languages. However, creating consistent and correct models can be difficult due to the inherent complexity and hidden dependencies. We propose a metamodel, consistency and correctness criteria for fragment-based case management (fCM) [9]. fCM models are highly modular and capture data-centric processes concisely; however, they contain hidden dependencies, e.g., the behavior of the process is constrained by cardinality constraints in the data model. The criteria that we propose can be used to assist modelers to avoid mistakes.

In the next section, we introduce fCM with a metamodel and an example. Based on this, we present consistency and correctness criteria (Sect. 3). In Sect. 4, we discuss related work. Finally, we conclude and discuss our contribution (Sect. 5).

2 fCM Metamodel and Example

As depicted in Fig. 1, an fCM case model consists of a data model (*domain model*), a process model (*fragments*), and a goal specification called *termination condition* [9]. The data model consists of classes and (binary) *associations* among them[1]. Each class has an *object life cycle* (OLC), which defines the behavior of respective objects. A dedicated *case class* denotes the central object which is created exactly ones for each case.

An OLC is a finite state transition system. For each class, a set of *states* and state *transitions* is defined. This *object behavior* is instantiated by the process: knowledge workers execute activities that create data objects and update their states accordingly.

Activities are included in small control flow graphs, the so-called fragments. Activities read and write data objects that belong to a single input- and output-set, respectively.

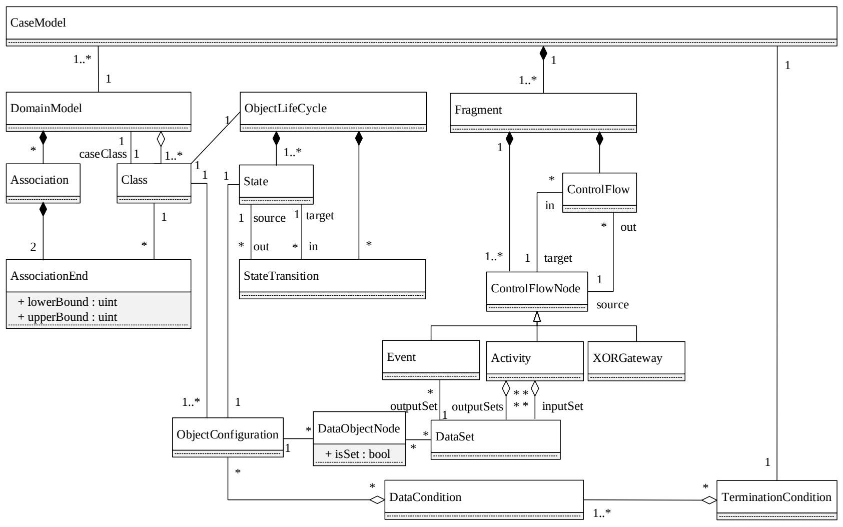

Fig. 1. fCM metamodel: a case model comprises a domain model including classes and associations, a set of fragments, and a termination condition.

[1] Due to space limitation, we do not consider *goal cardinality constraints*, which have been introduced in [6, 8].

Such objects are specified by *data object nodes*, which consist of an *object configuration* specifying the class and the state of the object and of an indicator (*isSet*, depicted as |||) showing whether multiple object's of the same type in the same state may be read.

Some fragments have start *events* marking the beginning of a new case. Such an event can create data objects. Also, a fragment can branch conditionally (*XOR-Gateway*).

While a fragment is similar to an imperative process model, all fragments operate on shared data. Thus, data requirements of activities can be used to model dependencies among fragments declaratively. At run-time, knowledge workers can instantiate and execute fragments repeatedly and concurrently if the activities' data-requirements permit.

A case that meets the termination condition can be closed by the knowledge workers.

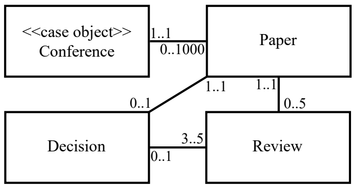

Fig. 2. Domain model of the paper submission and reviewing phase.

For an example, consider the paper submission and reviewing at an academic conference. The domain model Fig. 2 comprises the classes `Conference`, `Paper`, `Review`, and `Decision`. `Conference` is the case class. For each conference, multiple papers can be submitted; for each paper, multiple reviews can be created; for each paper, a decision is made based on at least three reviews.

Figure 3 shows the OLCs for the classes in Fig. 2. All but one OLC are described by a sequence of states, but OLCs can branch in case of alternative state progression and even contain disconnected subgraphs in case of alternative initial states. A decision object can be in one of the two alternative states *accepted* and *rejected*. OLCs do not define initial or final states. Since activities, create and change data objects, initial and final states are encoded in the process behavior. Adding them to the OLCs would add redundancy but provide only little value.

The case behavior is defined by a set of fragments (see Fig. 4). The example has one start event in fragment *f1*. When the start event occurs, a new case is started; the conference object is created; and activity "open submission" is enabled. Afterwards, the requirements of "submit paper" in fragment *f2* are satisfied. It can be executed repeatedly. Eventually, the knowledge workers perform "close submission" (*f1*) changing the state of the conference object and disabling fragment *f2* consequently. It also changes all papers

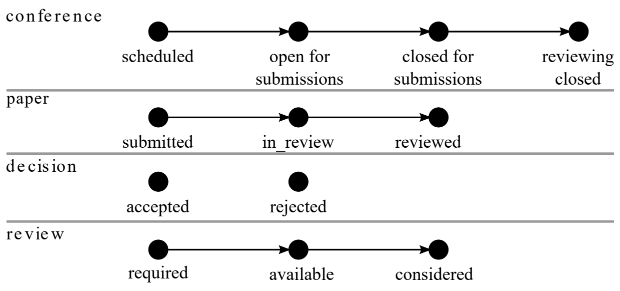

Fig. 3. Object life cycles for the classes in the domain model (Fig. 2)

to state *in review* (shown by the set indicator |||). Fragment *f3* can hence be executed to assign reviewers to papers. From the domain model, we know that each paper has at most 5 reviews. Therefore, fragment *f3* can at most be executed 5 times for each paper. As soon as a reviewer has been assigned, the review can be created (fragment *f4*). While there is a dependency between the fragments *f3* and *f4*, a review can be created before, after, or during the assignment of other reviewers. The order is determined by the knowledge workers. Similarly, activity "decide on paper" (fragment *f5*) can be executed as soon as all reviews assigned to a particular paper have been created. The activity has three possible outcomes: if there are less than 5 reviews for the paper, an additional reviewer may be assigned (output set {review[required]}; if there are at least 3 reviews, the paper may be rejected (output set {paper[rejected],reviews[considered],paper[reviewed]}) or accepted (output set {paper[accepted],reviews[considered],paper[reviewed]}). The different output sets are not part of the visual notation. Once all papers of the conference are in state *reviewed*, the reviewing can be closed (fragment *f1*).

The termination condition `conference[reviewing closed]` is satisfied after fragment *f1* has been executed completely. The case can be closed.

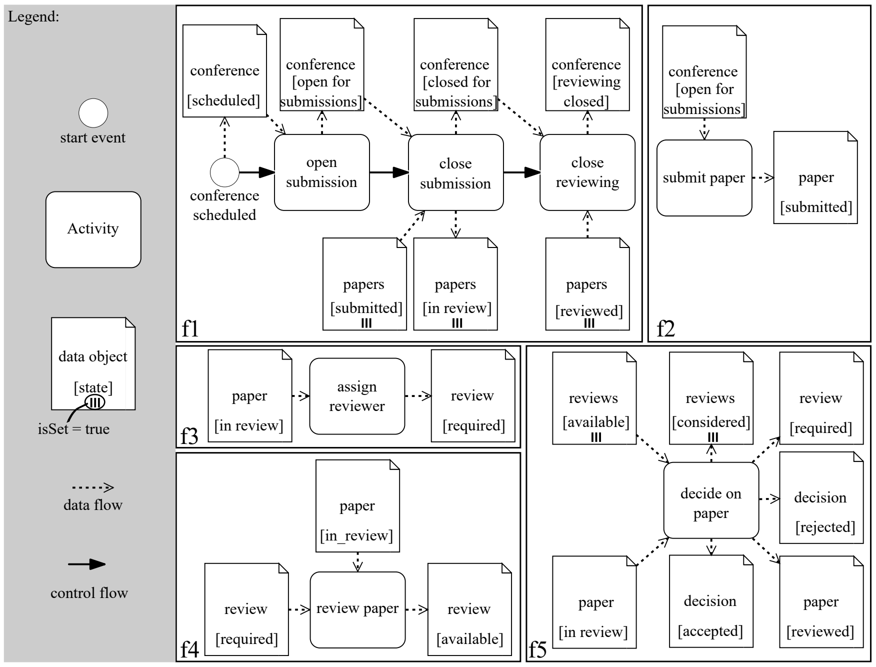

Fig. 4. Fragments for the paper submission and reviewing at an academic conference. Fragment f1 captures the progression of the conference. Fragment f2 handles the paper submission, f3 the assignment of reviewers, f4 the creation of reviews, and f5 the decision whether a paper is accepted, rejected, or an additional review is required.

3 Consistency and Correctness Criteria

As imminent from the example, all parts of the case model play together during a case. Fragments are connected through shared data, whose structure and behavior is modeled by the domain model and OLCs. The knowledge workers choose from enabled activities to progress the case towards the termination condition. Therefore, it is important that i) the parts are correctly modeled and ii) consistently integrated. In this section, we briefly sketch some structural correctness and consistency criteria that must be satisfied.

Assumptions. While the domain model focus on structuring the data, associations have behavioral implications. For one, they may define a partial order in which objects must be created [15]. In the example, every review requires a paper, but a paper may have no reviews (yet). Consequently, the review cannot be created before the paper. However, such an order can only be inferred if one object depends on another. Therefore, we require that all associations are existential: at least one of the corresponding lower bounds must be positive. Furthermore, we only allow one association between a pair of classes and disallow many-to-many associations (one of the association's upper bounds must be 1). If these assumptions are satisfied, new associations are only established when new objects are created, e.g., activity "assign reviewer" reads a paper and creates and associates a review. If the assumptions are violated, the domain model can be reified: new classes can be introduced in place of the violating associations.

Additionally, we make assumptions about the structure of fragments. We assume that fragments are acyclic. This does not limit the expressiveness since loops can be resolved into repeatable fragments. For example, in a purely imperative process model, activity "submit paper" would be part of a loop instead of a fragment. We furthermore assume that each fragment is either initial or non-initial. This means they either start with an event (e.g., fragment *f1* in Fig. 4) or with an activity (all fragments but *f1*)—never both.

Consistent I/O Behavior. Since fCM models consist of data and behavioral parts, most dependencies apply to the I/O behavior of activities. Correct I/O behavior depends on the domain model, the object life cycles, other activities, and even the termination condition.

First, all data requirements must be *satisfiable*. Therefore, some activity must produce the object configuration (object in a certain state) required by other activities or the termination condition. This means, each data configuration used in a data object node or a data condition should be referred by a data object node that takes part in at least one output set. For example, the object configuration `conference[reviewing closed]` must be written by at least one activity, i.e., "close submission". Assuming the termination condition would instead be `conference[proceedings published]` and the state would be an allowed successor of *reviewing closed*, the termination condition would be insatiable since no activity writes the conference object in the respective state.

Furthermore, for each output set of an activity must exist a respective input set so that the combination conforms to the constraints of the OLCs and the domain model.

A valid input-output-set combination is *OLC conform* [9]: all the subsumed state transitions must be present in the OLCs. If an activity "skip submission" changes the state of conference from *scheduled* to *closed for submission*, the OLC would be violated. This property is called *object life cycle conformance*.

Next, each object that is created requires a specific *context*, a set of objects it depends on. The required context is defined by the existential associations. In the example, the decision requires a paper and three to five reviews. This context must be provided by the input-output-combination. The required objects must either be read or co-created. If activity "assign reviewer" would not read a paper object, the requirements for the review would be violated. If execute anyway, the created review object would violate the cardinality constraints specified in the domain model (cf. Fig. 2).

What about the reviews required for a decision? According to the cardinality constraints, a decision existentially depends on three to five reviews. In such cases a set of objects (`isSet=true` visualized by |||) must be read, e.g., a set of at least three reviews must be read to create a decision. We call this *mandatory batch behavior*.

Finally, if a set of objects is read, it must be clearly defined. Therefore, each input set that contains a data object node with `isSet=true` must also contain a data object node with `isSet=false` for an associated class. The respective object is used to determine the set of objects that is co-read when executing the respective activity. In the example, "decide on paper" reads all reviews that belong to the paper. Without the paper, the set cannot be determined from the context.

The presented criteria are not domain specific, i.e., they do not only apply to the example fCM model but to all possible fCM models. Any case model that satisfies all criteria presented in this section is *structural consistent*. The structure of one part (e.g., the fragments) does not contradict the structure of other parts (e.g., the domain model).

4 Related Work

The fragment-based case management [9] approach is a production case management approach based on [11]. A metamodel for fCM has been proposed in [5]. The metamodel is close to [9] and includes elements that are fix for all models, such as generic life cycles for activities. Additionally, extensions have been proposed that consider associations [7] and cardinality constraints [6, 8]. The metamodel presented in this paper is the first fCM metamodel considering associations and cardinality constraints.

Besides fCM there are other approaches combining information from process and data modeling. Combi et al. [2] and Meyer et al. [12] combine data models, i.e., UML class diagrams, and process models, i.e., BPMN diagrams. [2] describes and detects inconsistencies while [12] derives SQL-queries for enactment. Montali et al. [13] introduce DB-nets—a Petri net-based formalism for modeling data-base accessing processes that adhere to data constraints (e.g., primary key, foreign key, and cardinality constraints). Therefore, they introduce a transaction mechanism to the processes. Ghilardi et al. [4] present catalog-nets to formally model processes with access to read only data-bases. While these approaches elaborate the connection between data and process, they are purely imperative and not suited for knowledge-intensive processes. However, DB-nets and catalog nets are interesting formalisms, which may be suited to formalize fCM's semantics and to define/verify behavioral correctness notions.

Other case management approaches exist. The two most prominent ones are the Guard Stage Milestone [10] approach and the derived Case Management Model and Notation [14] standard. Both approaches arrange activities into stages and compose behavior based on (data-based) pre- and post conditions. While the approaches assume a data model (called information model), they do not specify how it is modeled and integrated into the process specification.

5 Conclusion

Case models capture knowledge-intensive processes, which are often data-driven, multi-variant, and non-repeatable. Consequently, models become quite complex as they must integrate data and flexible processes. In this paper, we present a metamodel, consistency and correctness criteria for fCM models. All fCM models must satisfy these criteria.

Future work may elaborate them. First, formal definitions should be provided, e.g., using first-order logic or the object constraint language. Such definitions can be the base for implementing i) a verification tool that detects violations of the criteria and ii) a modeling tool that support case designers. Such a tool can highlight violations and offer auto-completion/correction. For example, when a case designer models an activity that creates a data object, all required objects (according to the domain model) can be added to the activity's input set if they are created by other activities or output sets otherwise.

Structural correctness and consistency is important and verification is computational in-expensive compared to state space analysis. However, such criteria cannot guarantee correct behavior. In future work, we want to investigate behavioral correctness criteria that apply to case models. A first example is weak termination: in the initial state, it should be possible to reach a state that satisfies the termination condition.

Furthermore, fCM does not capture all aspects of a case. Knowledge-intensive processes are also about knowledge workers, their rights, capabilities, and collaborations among them [3]. In the future, the fCM metamodel and language may be extended to account for the user perspective. The correctness and consistency criteria may subsequently be refined to consider the additional information.

Nevertheless, we believe that the presented metamodel and criteria can lead to tool support for fCM that may ultimately improve the accessibility of the fCM approach.

References

1. van der Aalst, W.M.P., Weske, M., Grünbauer, D.: Case handling: a new paradigm for business process support. Data Knowl. Eng. 53(2), 129–162 (2005)
2. Combi, C., Oliboni, B., Weske, M., Zerbato, F.: Conceptual modeling of processes and data: Connecting different perspectives. In: Conceptual Modeling - 37th International Conference, ER 2018, Xi'an, China, October 22-25, 2018, Proceedings. pp. 236–250 (2018)
3. Di Ciccio, C., Marrella, A., Russo, A.: Knowledge-intensive processes: Characteristics, requirements and analysis of contemporary approaches. J. Data Semant. 4(1), 29–57 (2015)
4. Ghilardi, S., Gianola, A., Montali, M., Rivkin, A.: Petri nets with parameterised data - modelling and verification. In: Business Process Management - 18th International Conference, BPM 2020, Seville, Spain, September 13-18, 2020, Proceedings. pp. 55–74 (2020)
5. Gonzalez-Lopez, F., Pufahl, L.: A landscape for case models. In: Enterprise, Business-Process and Information Systems Modeling - 20th International Conference, BPMDS 2019, 24th International Conference, EMMSAD 2019, Held at CAiSE 2019, Rome, Italy, June 3-4, 2019, Proceedings. pp. 87–102 (2019)

6. Haarmann, S., Montali, M., Weske, M.: Technical report: Refining case models using cardinality constraints. CoRR abs/2012.02245 (2020), https://arxiv.org/abs/2012.02245
7. Haarmann, S., Weske, M.: Correlating data objects in fragment-based case management. In: Business Information Systems - 23rd International Conference, BIS 2020, Colorado Springs, CO, USA, June 8-10, 2020, Proceedings. pp. 197–209 (2020)
8. Haarmann, S., Weske, M.: Data object cardinalities in flexible business processes. In: Business Process Management Workshops - BPM 2020 International Workshops, Seville, Spain, September 13-18, 2020, Revised Selected Papers. pp. 380–391 (2020)
9. Hewelt, M., Weske, M.: A hybrid approach for flexible case modeling and execution. In: Business Process Management Forum - BPM Forum 2016, Rio de Janeiro, Brazil, September 18-22, 2016, Proceedings. pp. 38–54 (2016)
10. Hull, R., Damaggio, E., Fournier, F., Gupta, M., Heath III, F.F.T., Hobson, S., Linehan, M.H., Maradugu, S., Nigam, A., Sukaviriya, P., Vaculín, R.: Introducing the guard-stage-milestone approach for specifying business entity lifecycles. In: Web Services and Formal Methods - 7th International Workshop, WS-FM 2010, Hoboken, NJ, USA, September 16-17, 2010. Revised Selected Papers. pp. 1–24 (2010)
11. Meyer, A., Herzberg, N., Puhlmann, F., Weske, M.: Implementation framework for production case management: Modeling and execution. In: 18th IEEE International Enterprise Distributed Object Computing Conference, EDOC 2014, Ulm, Germany, September 1-5, 2014. pp. 190–199 (2014)
12. Meyer, A., Pufahl, L., Fahland, D., Weske, M.: Modeling and enacting complex data dependencies in business processes. In: Business Process Management - 11th International Conference, BPM 2013, Beijing, China, August 26-30, 2013. Proceedings. pp. 171–186 (2013)
13. Montali, M., Rivkin, A.: From DB-nets to coloured Petri nets with priorities. In: Application and Theory of Petri Nets and Concurrency - 40th International Conference, PETRI NETS 2019, Aachen, Germany, June 23-28, 2019, Proceedings. pp. 449–469 (2019)
14. (OMG), O.M.G.: Case management model and notation (CMMN) (December 2016), https://www.omg.org/spec/CMMN
15. Snoeck, M.: Enterprise Information Systems Engineering - The MERODE Approach. The Enterprise Engineering Series, Springer (2014)
16. Swenson, K.D.: Position: BPMN is incompatible with ACM. In: Business Process Management Workshops - BPM 2012 International Workshops, Tallinn, Estonia, September 3, 2012. Revised Papers. pp. 55–58 (2012)
17. Swenson, K.D.: State of the art in case management - 2013 (2012), https://www.aiim.org/PDFDocuments/CaseManagement2013.pdf

All links were last followed on January 18, 2021.