2018 IEEE 22nd International Enterprise Distributed Object Computing Conference

A Comparison of Flexible BPMN and CMMN in Practice: A Case Study on Component Release Processes

André Zensen
Bielefeld University of Applied Sciences
Bielefeld, Germany
Email: andre.zensen@fh-bielefeld.de

Jochen M. Küster
Bielefeld University of Applied Sciences
Bielefeld, Germany
Email: jochen.kuester@fh-bielefeld.de

***Abstract*—The Object Management Group maintains two notations to model business processes, BPMN and CMMN. While the two follow different approaches, both offer structures to model flexible processes or parts thereof. This gives rise to the question which standard should be chosen to adequately model such processes. We compare BPMN with a focus on its ad-hoc sub-process with elements of CMMN case models along a practical case study. While BPMN offers a certain degree of flexibility, CMMN has several benefits but also drawbacks. We discuss advantages and disadvantages of both notations. To answer the question of which notation to use for modeling flexible processes, we derive simple guidelines to help in making an informed choice.**

***Keywords*-Business Process Management, BPMN, Case Management, CMMN, Case Study, Data-driven Processes, Knowledge-intensive Processes**

I. Introduction

The Object Management Group (OMG) maintains two notations for modeling processes, the Business Process Modeling Notation (BPMN) and the Case Management and Modeling Notation (CMMN). While BPMN [2] has established itself for modeling highly structured processes, CMMN [6] is a recent addition meant as a complementary notation to BPMN with a focus on knowledge-intensive processes [5]. It follows concepts of case management [23], [24]. As an expression of these, it provides different elements and execution semantics compared to BPMN to model flexible processes. BPMN also offers a structure to model such processes, the *ad-hoc sub-process*. The question arises which notation provides a better fit for modeling flexible processes or parts thereof.

To make an informed choice, we conducted a practical case study. For the study, we model the process employed by a major company to release components used in high-end domestic appliances and commercial equipment. We use an imperative and a declarative approach. The process shows both routine and flexible parts. It can be characterised as a case of production case management (PCM) [23], [17]. To compare the approaches for the process at hand, we use BPMN and CMMN to cover differing degrees of flexibility [19], from structured to unstructured and unforeseeable in terms of routing.

We discuss the advantages and disadvantages of both approaches by highlighting parts of the process, before we derive simple guidelines. While the findings can not be used to make generalized statements, they can aid in choosing which standard to use to model processes or parts of processes requiring flexible parts.

The work is structured as follows: Section II introduces the approaches and modeling standards. Section III describes the case study and process. Section IV describes the models created to capture the work on tasks needed to release a component for production. A comparison is made in section V to highlight advantages and disadvantages. Derived guidelines are given in section VI. Section VII presents related work with a focus on CMMN modeling, before section VIII concludes this work.

II. BPMN and CMMN for Process Modeling

The Business Process Model and Notation is a successor to other imperative process modeling languages or notations, such as Petri nets [25] and Event-driven Process Chains (EPC)[20]. Figure 1 shows a subset consisting of the elements used for the case study. A pool contains a typed start event, connected to an activity with a sequence flow. Another sequence flow leads to an expanded ad hoc sub-process containing a collapsed one. It uses and generates data objects. A parallel gateway splits the sequence flow inside to trigger an outgoing (thrown) and incoming (caught) message event. An end event concludes the process. Execution semantics are further described in section IV-A.

Combining a diverse, formalised graphical notation with execution semantics in an interchangeable standardized format, it has since its release in late 2011 become an important standard for business process modeling. The execution follows strict and explicitly modeled paths, defined by *flow objects*, such as *activities* linked by *connecting objects* defining sequences. Branching and routing is made possible by *gateways* utilizing conditions and expressing decisions. Data is handled as input and/or output of single *activities*. An exception to the regular strict pathways is the *ad hoc sub-process*, which is marked with a tilde symbol at the bottom of the *sub-process activity*.

A different approach is the declarative Case Model and Management Notation, first released in late 2016. Heavily influenced by an artefact-centric view of processes [18], the Guard-Stage-Milestone approach proposed in [10] and the

2325-6362/18/$31.00 ©2018 IEEE
DOI 10.1109/EDOC.2018.00023

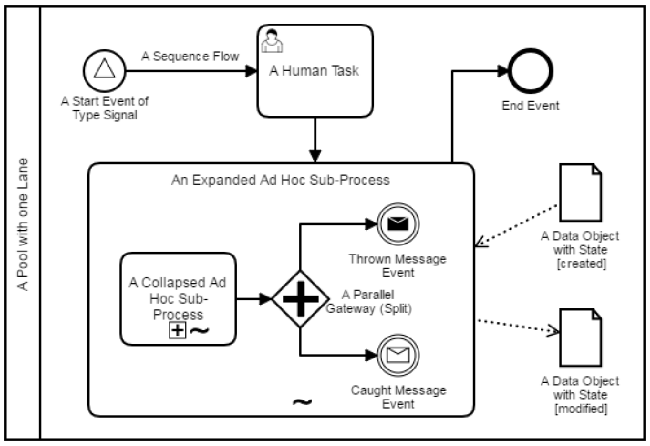

Fig. 1. Used BPMN Elements

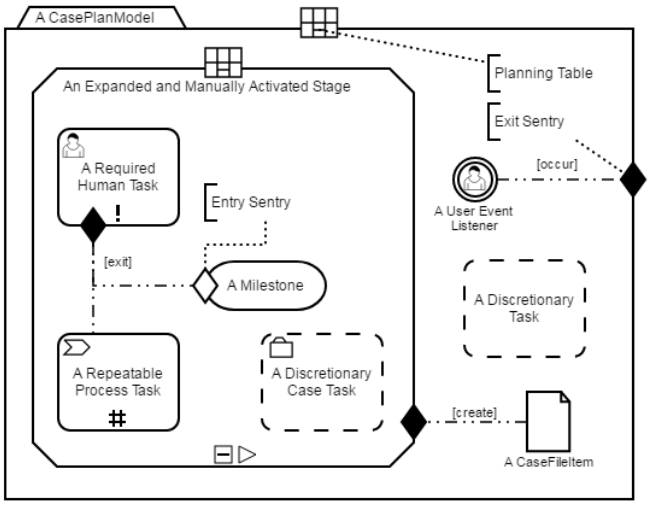

Fig. 2. CMMN Elements

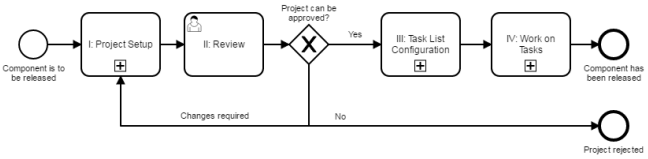

Fig. 3. High level overview of the component release process

case handling paradigm described in [26], it aims to become a standard for case management. It gives new importance to entity life-cycles and flexibility in execution of processes. Research published on CMMN covers different aspects, such as its fit for case management definitions [14], [15] and adaptive case management in particular [12].

Complexity metrics [16], support for established workflow patterns in comparison to BPMN [7] and prototypical implementations to support case management based on CMMN [13] are topics of other research. Recent publications also focus on CMMN to model case study scenarios. An overview is given in the related works in section VII.

Figure 2 shows the main elements available in CMMN. *Tasks* can be typed and annotated with *decorators*, e.g. with an exclamation mark when they are required, which is not available in BPMN. *Sentries* can be used as entries and exits to control flow options and linked to life cycles of elements, such as the creation of a document or the completion of a task.

Like BPMN, it combines formalised graphical elements and execution semantics in an interchangeable standard. Using a small subset of BPMN elements and several new elements including a *process task*, it introduces new concepts to support case-oriented processes. New and old elements have defined life-cycles which interact with each other. They influence which elements transition to an enabled state to be activated by case workers.

Pathways are not imperatively defined beforehand as in the majority of BPMN constructs, but can change depending on life-cycles, conditions as well as user choices. *Sentries* can be attached to elements such as *stages*, *tasks* and *milestones*. *Sentry criteria* can be defined using life-cycles and other defined conditions to determine when an element is enabled to be selected, activated and worked on. To further support flexibility, a planning phase and *discretionary items* are introduced. During planning, discretionary elements such as *tasks* can be chosen to be enabled, or not.

A new *milestone* element underscores a goal oriented approach: a 'what is the overall goal' takes precedence over the explicit and imperative 'how to reach the goal' of a process and execution of an instance.

III. Case study: component release process

We conducted the case study to analyse and model the process in BPMN and CMMN for the purpose of documentation, management and improvements. The models were compared to see which OMG standard provides a better fit for the whole process and certain parts of it.

The release of components, which can be a single component such as an electrical relay or a combination of components like a touch screen display, is handled in projects. The overall process can be divided into four parts: (I) a project set up, (II) approval or rejection of it, (III) a variable task list configuration from a pool of default tasks, and (IV) the work on the tasks along six logical steps. The task lists change depending on the component family and project lead who configures them.

Figure 3 shows a simplified high level overview of the four process steps. Tasks for one component family, *relays*, are used for the models described in section IV.

If a new project for component release is approved, a supplier manager chooses tasks for the project from a configurable default list of tasks. The tasks are scheduled, staff assigned to them and notified. Some tasks are optional, e.g. when a component is already specified and catalogued, or a prototype can be build internally.

The configured task list for the component family *relays* can include up to six steps to completion. Technical specifications

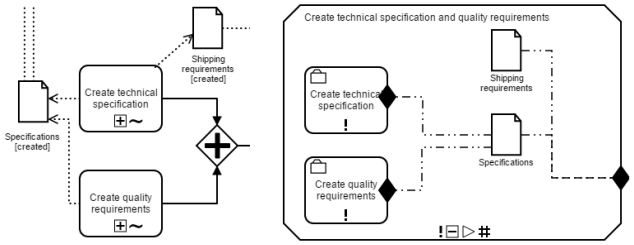

Fig. 4. Ad hoc sub-processes and case tasks used to hide complex work in BPMN (left) and CMMN (right)

and quality requirements for the component are created by hardware developers and the construction department (1). Based on the specifications and shipping requirements, a tender for supply is issued by a purchaser and the documents loaded into a work-flow in the used enterprise resource planning system (ERP [22], [11]). A supplier is chosen. Design specifications are discussed and synchronised with the supplier who eventually produces the component, often in large quantities (2). A number of prototypes is build or procured and reviewed internally (3).

Final specifications are created and a A0 series is procured. The series is sampled and, if required, a feedback loop with the supplier is initiated. Final performance specifications are created for a B0 series (4).

The series is sampled like before (5). In parallel, different capabilities of the supplier have to be reviewed (6), such as being able to produce in series, at a specified rate and within quality requirements. Some tasks and evaluations are optional for catalogued material components. Long-running life span tests are performed on the component.

Once the B0 series can be produced in accordance with the underlying specifications, the component is released into the ERP system and the project is completed. If a specification detail changes to a certain degree, several of the steps and contained tasks have to be repeated. Documents are adjusted while retaining the chosen supplier. The partially optional work and rework require flexibility.

IV. Modeling with BPMN and CMMN

The process is modeled in BPMN as an *ad hoc sub-process*. In CMMN, the task list is modeled as a *case model*. Both include the six steps described in section III. Other component families share similar or identical tasks with the modeled task list. For brevity, not all data artefacts have been modeled in the models, e.g. different drawings, calculations and other attachments used to create the specifications are subsumed in one data object. Several complex work items in collapsed *sub-processes* or *case tasks* (see Fig. 4) are not further shown, but can be considered relatively unstructured and knowledge-intensive work. An example of such a complex work item is the procurement of a prototype, which involves the departments of research and development, production, logistics and finances.

A. Modeling with BPMN

Figure 5 shows the chosen *ad hoc sub-process* to support flexible execution, e.g. the ability to skip certain tasks or go back to others. Regular, imperative BPMN structures do not offer such a degree of flexibility.

Initially, all *activities* without incoming *sequence flows* or constrained by *data associations* are available for execution and can be activated manually. *Sequence flows* and *data objects* restrict the available *activities*. Logical dependencies are expressed with *sequence flows*, e.g. the procurement of a prototype.

The predominantly *human tasks* are partially structured by *sequence flows*. One *parallel gateway* is used to join the specifications created in two *sub processes*. *Intermediate message events* are used as feedback messages for synchronizing the specifications with a chosen supplier.

The process relies on several *data objects* which are for the most part modeled with the state *final*. One more elaborate example of different states is shown with regards to task *Synchronize specifications with supplier* in Fig. 6. The *data object Design specifications* is shown in the states *created*, *modified for review* and *reviewed*. Its final state is implicitly modeled with the *data object Prototype specifications*. The collapsed *sub process Start feedback loop with supplier* is mandatory after both the prototype and A0 series samples have been procured. Explicit routing is not used here to keep it flexible and reusable. In the context of the A0 series, the *data object Final performance specifications [final]* is created and used for the evaluations and test.

The evaluation tasks and life span test produce different *data objects* documenting the individual results. Internally, during work on the *tasks*, the *data objects* can assume all the previously mentioned states before reaching the state *final*.

Once the *ad hoc sub-process* is completed, a status is sent and the component has been released. The completion condition is the completion of the *task Issue release in ERP*.

B. Modeling with CMMN

Figure 7 shows the *case plan model*, which supports flexibility in its specified semantics. It contains the six steps described in section III in six *stages*. *Milestones* convey the overall status and serve as *entry criteria* for the next *stage(s)* and *tasks*. The *stages* are required except for two *discretionary stages*. They are also repeatable for rework on changed specifications. *Task Issue release in ERP* completes the case and releases the component for production.

The manually (see the *decorator* at the bottom of the *stage* seen in Fig. 2) started entry *stage Create technical specification and quality requirements* is exited by completing the *case tasks* and by creating the individual specifications and shipping requirements. After exiting, the *milestone Specifications created* is completed.

The *stage Choose supplier and synchronize specifications* contains two *required* and one *repeatable discretionary task* for selecting a suitable supplier. A *process task* loads the specifications into an ERP process. Its completion is the *entry*

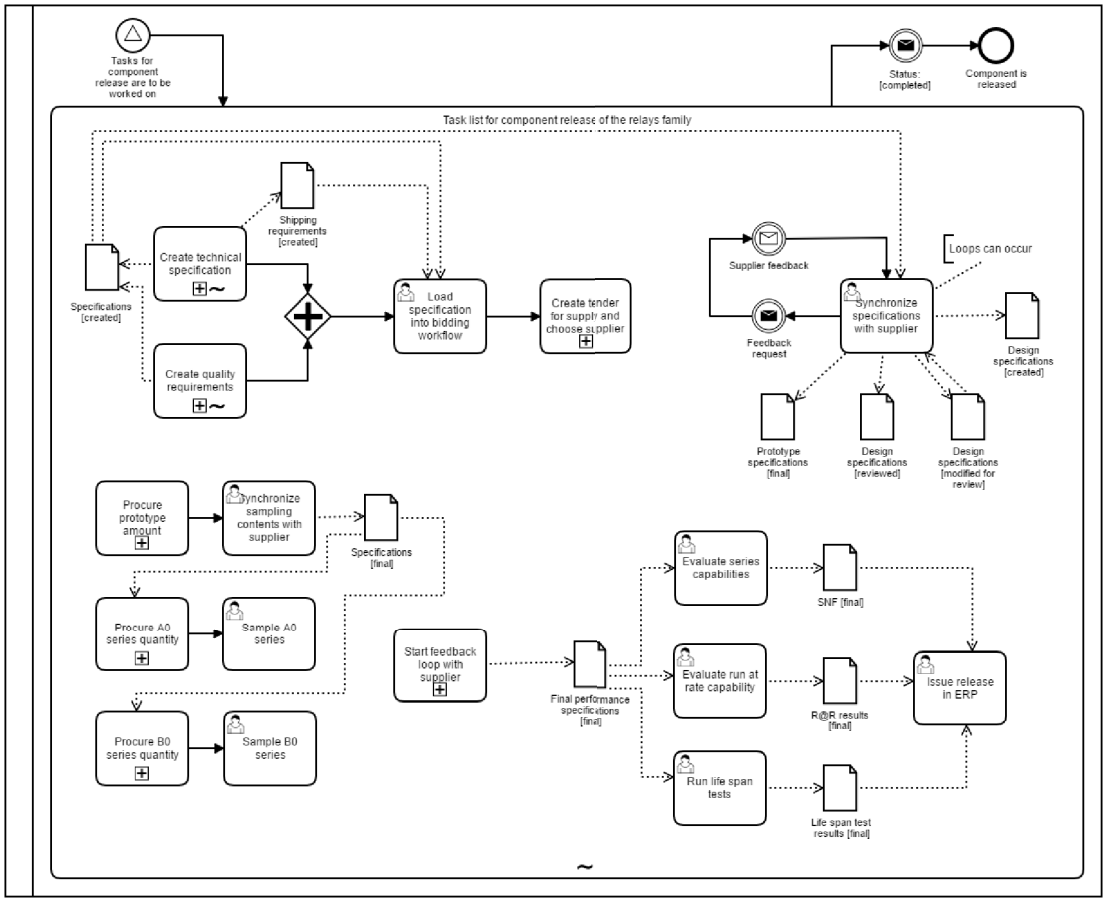

Fig. 5. Partly imperative ad hoc sub-process containing tasks

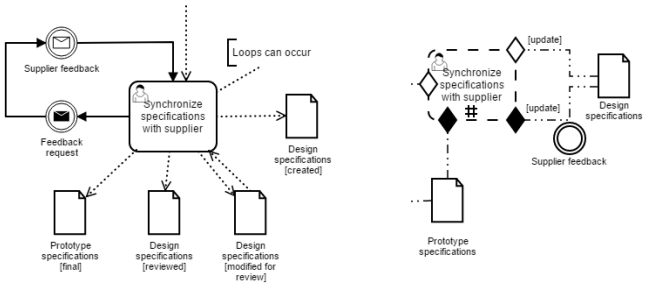

Fig. 6. Iterative work on a task to coordinate a final stage of a document in BPMN (left) and CMMN (right)

criterion for the synchronizations with the supplier and creation of design specifications. An *event trigger* is used to model a supplier feedback which updates the design specifications and creates a loop. Once the prototype specifications are created, the *task* and *stage* are exited and the *milestone* is completed.

If planned, *stage Procurement of prototype and creation of final specification* is enabled. Two *required process tasks* express procuring the prototype materials and a feedback loop. The *stage* is exited when both are completed. A *human task* used to synchronize the samples with the supplier is enabled. Its completion creates the *case file item Final specifications* and completes a *milestone*.

The stage *Procurement and sampling of A0 series* is enabled. Two *required process tasks* express the procurement and another feedback loop, similar to the one used in the prototype stage before. The underlying processes are not shown. The created final performance specifications serve as the *exit criterion* of the *stage*. It is the *entry criterion* for the completion of the *milestone* and the two following *stages*.

Similar to the A0 stage, *stage Procurement and sampling of B0 series* contains a *required* procurement *process task* and a *required human task* to sample the series. Once completed, the *stage* is exited and the *milestone* completed.

The other *discretionary stage*, *Proof of capabilities*, contains two *discretionary human tasks* which are required if planned. Their *exit criteria* are the created *case file items* documenting the evaluation results. Their creation is also the *exit criterion* of the *stage*.

The *required task*, *Run life span tests*, is outside the *discretionary stage* containing the evaluations, since it is is mandatory. Its *entry criterion* is the *milestone A0 series sampled*.

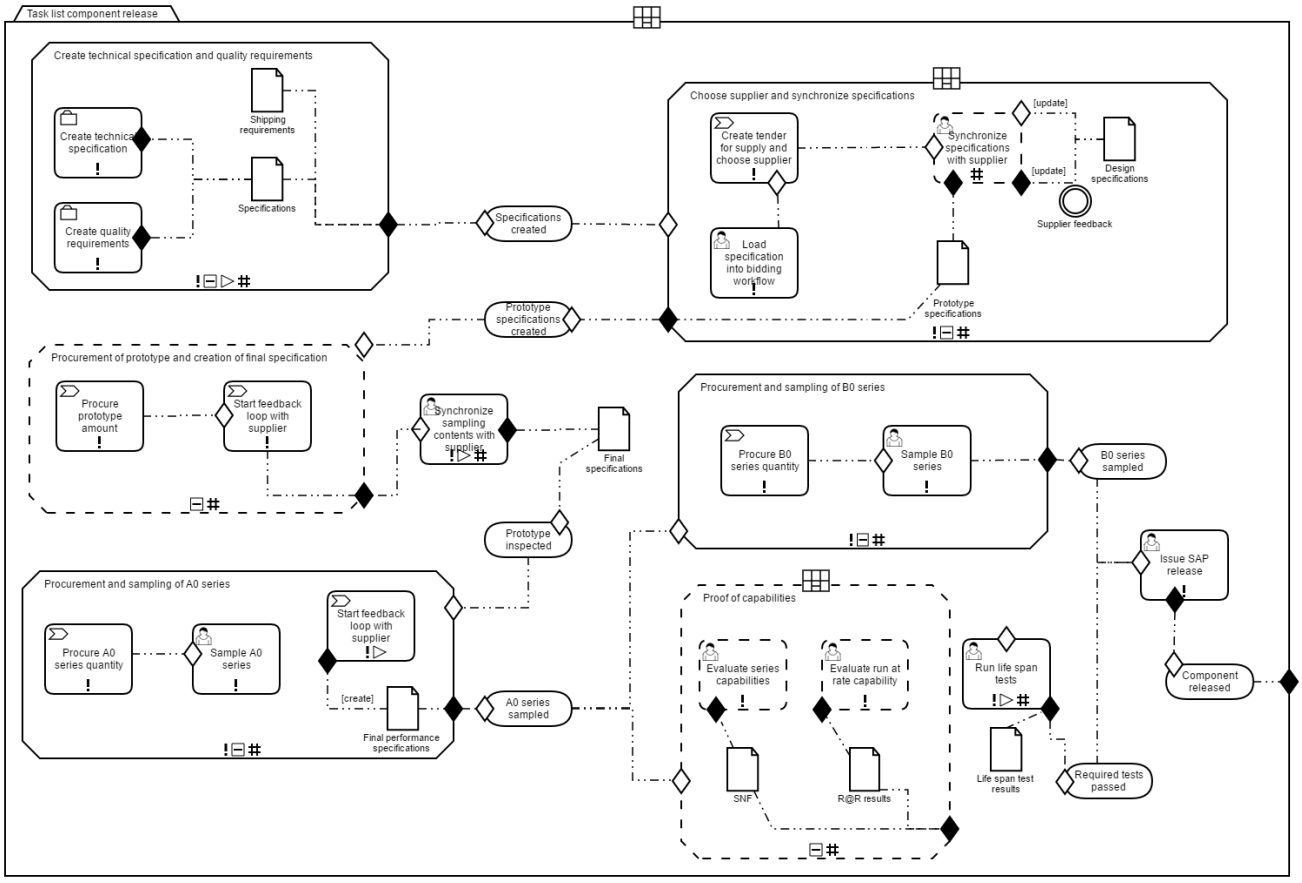

Fig. 7. CMMN task list case

Created *case file items* containing the evaluation results serve as the *exit criterion* and triggers the completion of the *milestone Required tests passed*.

After both *milestones B0 series sampled* and *Required tests passed* are completed, the *entry criterion* for the *required task Issue release in ERP* is fulfilled. Its completion exits the *stage* and triggers the last *milestone* of the task list, *Component released*. This concludes the process.

V. Comparison of Models: Advantages and Disadvantages

To highlight advantages and disadvantages of BPMN and CMMN, we use four categories:

Process structure includes overarching structural elements, e.g. *pools/lanes* in BPMN and *cases/stages* in CMMN. These are commonly used to shape a process, group tasks and define/express scopes/organizational units.

Routing and control-flow includes elements used to define sequences and/or routes to take during the execution of a process, e.g. *gateways* in BPMN and *sentries* in CMMN. It also covers path flexibility during process execution, e.g. whether tasks can be skipped or returned to at a later point and how to model such behaviour.

Communications and events includes elements used for communications inside a process (e.g. among structural elements and/or tasks), between *processes/cases* and those used to model events. It also includes the propagation of statuses and state transitions.

Data aspects and data-flow focus on elements used to model data and its flow in and between *processes/tasks*, including dependencies.

A. Advantages and Disadvantages of BPMN

a) Advantages: Concerning *routing and control-flow*, sequences can be directly seen and are visibly defined by the imperative *sequence flows* as seen in Fig. 6. BPMN offers different means of modeling *communications and events*. For messages in the conventional sense, the *send* and *receive task* as well as *message events* can be used to communicate and convey the overall progress of the *process*. As for *data aspects and data-flow*, inputs and outputs are displayed as *data objects* with annotated states and data flows. A more elaborate example is the different states of the design specification data used in the initial synchronization seen in Fig. 6.

b) Disadvantages: With regards to *process structure*, *pools* and *lanes* are not available in *ad hoc sub-process*. The

involved departments and thus roles/employees working on the tasks can not be modeled. The specification restricts elements to *activities* which 'MUST be used', while elements which 'MAY be used' are '**Data Object, Sequence Flow, Association, Data Association,** [...] **Gateway and Intermediate Event**'. Elements which 'MUST NOT be used' are *start* and *end events*, *conversation* elements and *choreography activities* (see [2, Sec.10.2.5, p. 182]). The *grouping artefact* may be used, but offers no execution semantics. Logically dependant tasks can not be grouped in flexible structures without introducing more elements such as *sub processes*.

Though a differentiation is not always possible due to the collaborative nature, certain tasks, e.g. those regarding the creation of specifications (*tasks Create technical specification* and *Create quality requirements* as seen in Fig. 6), could be mapped to a technical department. Alternatively, another *process* with *lane* structures could be called, but it would not offer flexible execution semantics.

Even only partially imperative *routing and control-flow* may become problematic: several tasks in the list can be optional and can be skipped, others might have to be performed at different times or points throughout the list and *sequence flow*. The flexible return to a *task*, e.g. after changes to a document, have to be modeled either repeatedly along the *sequence flow*, e.g. with *boundary events* branching off, in loop constructs (see Fig. 8) or remain underspecified and potentially available at all times.

Explicit routing can lead to 'spaghetti-flows'. The *sequence flows* used can not be skipped. To model this behaviour, more elements would have to be introduced or the sequences removed, leaving the workers with less guidance.

If explicitly modeled data flows with all states are used, e.g. to enable a *task* again after changes to the specification, the model becomes less readable. Even a reduced number of documents would double or triple the amount of *data artefacts* in the model. They would be needed to model each state of a document to display possible behaviour and conditions to return to a previously completed *task*.

It is unclear when available *tasks* should be selected inside an *ad hoc sub-process* without knowledge about the process, or control structures like additional *gateways* or *business rule tasks*. By default, all *tasks* with no incoming *sequence flow* are available. Phases as described in the six steps in III and seen in the CMMN model (see Fig. 7) is difficult without introducing more routing elements or *sub-processes*. Additional control structures like *gateways*, *business rule tasks* and routing have to be introduced in order to make only selective *tasks* available again after previous completion.

As for *communications and events*, the available *message events* are sufficient for the requirements, except for one: a status is to be conveyed after logical units of several tasks have been completed. An outgoing *message event* is used for status information after the *ad hoc sub-process* completes, being semantically closest to the intent. A more specific element is missing. *Message events* are restricted to normal flows or *activity* boundaries and can not be used as standalone events

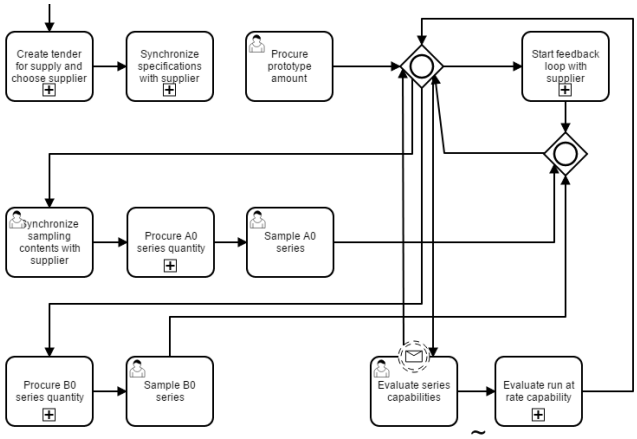

Fig. 8. Additional routes to express optionality in an ad hoc sub-process using a central OR-gateway

to signal a status, e.g. when a document is finalized.

With regards to *data aspects and data-flow*, the model contains a condensed amount of all identifiable documents. The reduced amount however does not include all state changes of the produced documents. Common states of the documents in the case study are *created*, *modified for review*, *to be reviewed*, *reviewed* and *final*.

Modeling all possible states of *data objects* and their respective flows for all possible *sequence flows*/routes decreases the readability of the model. The return to completed *tasks*, e.g. a document with the state *modified for review*, requires more data flow aspects. Fig. 8 hints at a possible construct using *inclusive gateways* to select specific *tasks* using conditions, in this case a reusable feedback task. *Data associations* also restrict flexibility, such as starting preliminary work of a task, as they make tasks wait until data is available in full.

A possible workaround to maintain readability is to break up the *process* into smaller ones along the *stages* in the CMMN *case model* (Fig. 7), which would make a significant overhead necessary. Control and communications aspects have to be connected with an overarching controlling process, managing data scope and flow among these.

B. Advantages and Disadvantages of CMMN

a) Advantages: The *process structure* of the model is defined by using *stages* to capture different areas or documents to be worked on, as well as the overall progression. In the *stages*, thematically similar *tasks* or those relying heavily on each other are grouped together and can be expressed as such.

Discretionary elements help to highlight optional work. Parts of the process can be skipped, depending on whether the required work or documents were created before and are available to be used or build upon. Individual planning of the *discretionary stages* and contained *tasks* further aids flexibility. As for *routing and control-flow*, routing among the *stages* is achieved with a combination of *sentries* and *milestones* as seen in Fig. 9, emphasizing the case progression in a step-wise fashion. *Sentries* and the *RepetitionRule*, which are evaluated

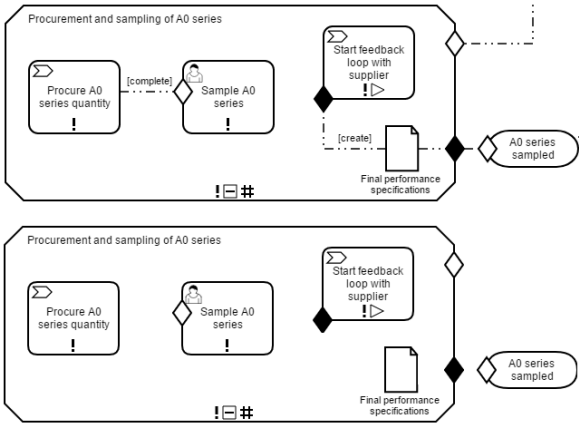

Fig. 9. Imperative routing in CMMN with sentries and milestones with connectors (top) and without connectors (bottom)

for *milestones*, *stages* and *tasks*, enable controlled, yet flexible selection depending on the defined *criteria* and conditions. They further allow deliberate jumps between tasks without being restricted by a strict sequence flow structure but conditions, e.g. of data (see [6, Sec.8.6.4, p.122ff.]).

Partially or completely imperative routings can also be modeled by using a combination of *connectors* and *sentries* as seen in Fig. 9. The *onParts* of *sentries* are used for imperative behaviour, highlighted by the use of *connectors* among *tasks*, *tasks* and *stages*, as well as *stages* and *milestones*.

Since *connectors* are purely visual elements with no execution semantics, they do not restrict routing options or control-flows. *Stages* can be active in parallel as required and single tasks can be executed in an ad hoc manner, such as feedback processes throughout the process at different times. Yet they clarify routes and control-flow aspects. For example, in conjunction with documents, they can be used in a similar fashion as *data artefact* annotations in BPMN. Unlike in BPMN though, data are first class elements, as their life cycles and thus statuses can directly control the flow without direct connections.

A special case is supported without having to rely on 'sequence-spaghetti' displaying all possible routes and constraints: when a specification detail underwent significant changes, large sections of the process encapsulated in the modeled stages have to be reviewed and worked on again. This dynamic behaviour, which is desired, can be expressed by CMMN with the use of *sentries* and *repeatable rules* together with the states of required *case file items*.

An advantage in terms of *communications and events* are *milestones*. The overall progress of the process can be explicitly conveyed. *Milestones* provide a semantically better fit compared to the BPMN modeling with *events*. They can also be used without any connections.

Another advantage of *events* in CMMN is the ability to connect a *user event* to *sentries* and *case file items*. Unlike explicit *sequence flows* leading to an *end event*, CMMN directly supports flexible state changes of a whole *case*. The project can transition from active to suspended at any time, i.e. be shelved when priorities change, and returned to at a later point.

An advantage with regards to *data aspects and data-flow* is to link *case file items* to *sentries*, *milestones* and *events* with *connectors* annotated with life-cycle states in order to express the dependencies (see Fig. 9. The information model of *case file items* is also specified, highlighting the role of data and documents in a *case*. While hierarchies can not be modeled graphically, the CMMN specification defines them.

The access to *case file items* throughout the process is also advantageous for data-driven processes. While the data might be restricted to individual roles, case workers can potentially access all documents, avoiding context tunneling which is described in [26] as a disadvantage of BPMN.

b) Disadvantages: The *process structure* of the CMMN model is built on *stages*. It does not clearly define roles, concepts like *pools* and *lanes* are not available. Roles can also not be mapped to *tasks* in a structural element used to represent the organizational units. *Stages* could be broken up to represent the involved staff of departments, but capturing the cross sections of these is not feasible, as they are not clear cut and many sub-stages would be required. *Stages* cannot be modeled to overlap graphically to highlight intersections. More specific roles are not defined in the CMMN specification.

The *routing and control-flow* without use of *connectors* and *sentries* can be difficult to understand (see Fig. 9. The task list is modeled using *connectors* and *sentries* to connect all elements with each other. Without these or additional annotations, deeper knowledge of the process progression and stage selection is required. *Sentry criteria* have to be defined but are not necessarily explicitly modeled with connections.

Sentries can also be attached to elements such as *stages* or *tasks* without any connections to other elements and without labelled *connectors* indicating the *required* states of a *task* or *case file item*. This can make it difficult to understand the model compared to an imperative sequence flow. When the criteria of a *sentry* attached to a *task*, *stage* or *milestone* are satisfied is not necessarily clear from the model without supporting elements as seen in Fig. 9.

Another aspect regarding *milestones* is scheduling. In its specification, BPMN nor CMMN offer a way for scheduling work and to capture differences in planned and actual times needed.

Iterations of tasks are also not as straight-forward as with BPMN and expressive control flows. Either no specific *sentry* element is attached to a *task*, except for the *repeatable decorator*, leaving out explicit criteria as to when the *task* is to be repeated. Or a *sentry* with an *entry criterion* is attached to a *task* in order to become enabled when the criteria of the *sentry* are satisfied.

Another option is to explicitly model the *entry* and *exit criteria*, e.g. to model a close loop as seen in Fig. 6. The creation and update states of the document act as the *exit* and *entry criteria* of the *repeatable task*. The shown *task* has two exits and the overall case progresses after the final document has been created, resulting in an exit of the *stage*. To model explicit incoming and outgoing communications, a *process task*

which encapsulates the communication *events* shown in Fig. 6 on the left would have to be used instead of the *event* seen on the right, which is used to signal documents sent and modified by the involved supplier.

Communications and events can not be expressed as in the BPMN model. CMMN only offers incoming/caught and not outgoing/thrown *events*, making only one-way communications available to model explicitly. Documents have to be reviewed internally and modified together with external partners, making communications in both ways necessary.

Simple notifications about the progression of the *case* or changed data as shown in Fig. 6 with regards to e-mail exchanges, or communications with other processes, cannot be modeled as easily as in BPMN. *Events* in CMMN are reactive and for receiving an *event*, not for outbound propagation beyond the *case*.

In order to model outgoing communications, *process tasks* have to be used, compared to a single element in BPMN. The *case* requires an underlying BPMN process with at least three elements, a *start*, *intermediate message* and *end event*, adding complexity to the model. Complexity is also added by the nested BPMN process.

Another aspect concerning events is scheduling of tasks with defined due dates, which are planned in advance in the case study. These are used for monitoring and project management apart from *milestones*. CMMN does not specify scheduling aspects for *tasks* nor *milestones*.

While *data aspects and data-flow* play a more important role in CMMN than in BPMN, it is unclear how the intricacies of a *case file* and the *case file items* contained can be included in the model or handled in regards to control-flow. *Sentry criteria* contain an *ifPart*, which generally reference the state of a *case file item*, but not specific values in a document. Thus the *case file items*' content and values have to be linked to its state. The definition and attribute *structuralRef* of a *case file item* can be used, but the life-cycle states of a *case file item* do not reflect properly on this. A *case file item* defined in the standard is either created, updated or discarded. A state reflecting on completeness of a *case file item* and thus a document or its absolutely necessary parts to progress the *case* is missing.

Also, hierarchies among *case file items* can not be expressed graphically. Several accompanying documents, such as specification drawings and notes, are to be attached to work of a *task*. These have been subsumed, e.g. as a single *case file item Specifications*, as seen in Fig. 6, hiding this information.

C. Comparative summary of advantages and disadvantages

Table I summarises the advantages and disadvantages modeling the task list work with BPMN and CMMN. A '+' indicates a clear advantage, a '+/-' partial advantages, and a '-' disadvantages. Partial advantages means the notation can be used with drawbacks, such as with elements used in a semantically different way. It further shows highlighted subcategories for each of the previously defined categories, as discussed in the previous section.

	BPMN	**CMMN**
Process structure		
Organizational units	-	+/-
Task mapping to unit/role	-	+/-
Flexibility (ad hoc, variability)	+/-	+
Routing and control-flow		
Clearly defined paths	+	+/-
Clearly defined decisions	+	+/-
Flexibility (skip, return/repetition)	+/-	+
Communications and events		
Outbound/two-way communications	+	-
Progression checkpoints	+/-	+
Ad hoc interactions	+/-	+
Data aspects and data-flow		
Life-cycles as decisions/triggers	-	+
Data hierarchies	-	+/-

TABLE I
COMPARISON OF BPMN AND CMMN

Neither a BPMN *ad hoc sub-process*, nor a CMMN *stage* can be structured using *lanes*. *Stages* can be used as a single unit or role to group tasks together. Variability in BPMN has to be expressed with additional imperative routings for each variation, unless such guidance is not desired. CMMN directly supports variability with discretionary elements.

Paths and decisions can be clearly defined in BPMN. *Gateways* structure depict possible routes and the decisions leading to them. Skipping *tasks*, or returning to single *tasks* is partially supported by making use of explicitly modeled routes. CMMN execution semantics directly support highly flexible task selection based on the user activating enabled *tasks*, *sentry criteria* which take into account life-cycles of elements, as well as *plan item controls* without being restricted to imperative *sequence flows*.

BPMN directly supports incoming and outgoing communications and event propagation. CMMN only directly supports inbound event propagation. Messages have to be sent using a *process task*. Displaying progression is directly supported by CMMN with its *milestone* element. In BPMN, *thrown (intermediate) events* can be used.

Ad hoc interactions, such as changing the state of the *process*/*case*, or triggering and reacting to other events, are possible in BPMN with *boundary events*. Due to its different and flexible execution semantics, CMMN supports such behaviour directly. CMMN uses life-cycles of *case file items* for *sentry criteria* and routing. It also specifies data hierarchies as *case file* with *case file items* as children.

VI. USING BPMN AND CMMN: GUIDELINES

Some guidelines can be derived from the previously described advantages and disadvantages. Generally, BPMN is best suited for routine work with little to no exceptions or required flexibility. For more flexible execution, CMMN seems to be a better choice than the *ad hoc sub-process* in BPMN. Both are meant to be complementary to each other, but each has its benefits with regards to aspects highlighted previously.

a) BPMN: is a good choice for work with little variation and optionality. Every deviation from a 'happy path' and guided

selection of tasks have to be modeled explicitly. Control flows and underlying decisions can be modeled comprehensibly. *Ad hoc sub-processes* can be used to add flexibility in parts of a process while still being bound to an imperative flow as a whole.

BPMN is strong on communication and events. Two-way communications can be modeled explicitly, being an important aspect of today's information-orientated work. Events cannot be used uncoupled from *tasks* or to act on *data objects*. *Events* can be used to directly trigger other *processes*.

b) CMMN: is best used if a high degree of flexibility and variation in terms of structure and routing is needed. Work can be actively selected and planned by the case workers. Routing can vary between strictly guided and completely left to the case worker. Optional work can be highlighted with *discretionary items*. *Stages* can be used to fragment a process' work into logical work units without explicit routing.

If data and reacting to state changes plays an important role, i.e. going beyond the usual BPMN data processing and usage of data inside *tasks*, CMMN is a good choice. Data is a first class citizen in CMMN and the defined data life-cycles can be used as triggers without being restricted to an imperative flow.

While events in CMMN are restricted to only three incoming events, they can be used uncoupled from a restrictive *sequence flow*. The *milestone* element is a useful addition to underline goal orientated knowledge work.

A CMMN *case model* could potentially include task list items required for similar component families. The required sub-set of *tasks* can be planned individually.

For highly structured processes, CMMN offers the *process task*, an explicit link to BPMN. The *process task* can be employed in areas where CMMN seems to be lacking: explicit two-way communications, automated tasks and routine work needed for a case to progress.

c) BPMN and CMMN: Both can be combined to cover the spectrum of very structured and necessarily flexible parts of a process model. For example, the project set-up and task list configuration of the case study process (see section III) is routine work and can easily be modeled in BPMN. Then, a configured case model in CMMN containing the task list items can be used to capture a more data-centric and flexible process part. Routine processes, which have also been modeled in BPMN but are not further shown, can be directly embedded as process tasks in CMMN as seen in Figure 10.

Alternatively, the BPMN process can include a CMMN case as a *call activity*, e.g. by replacing the last *sub process IV: Work on Tasks* in Figure 3 with the *case model* seen in Figure 7.

VII. Related Work

Breitenmoser and Keller compare BPMN with CMMN and focus on flexible structures and planning aspects of CMMN. They match elements of BPMN to those of CMMN, with findings similar to those highlighted in this work. Interestingly, they match *discretionary items* in CMMN to *ad hoc sub-processes* in BPMN. A literature review is given on different topics relating to both notations [3].

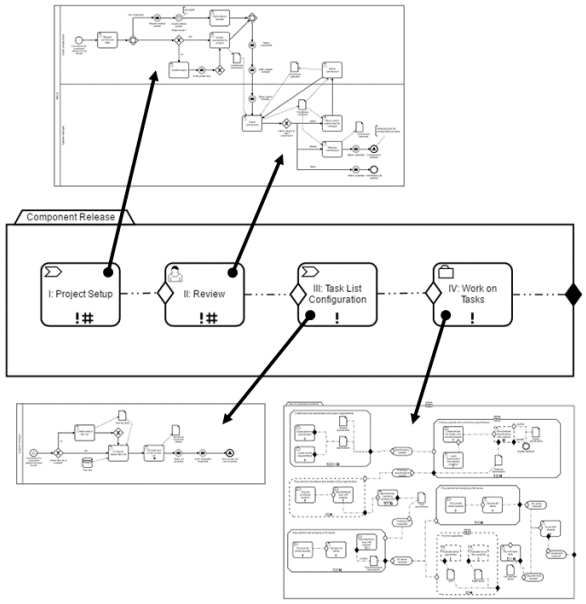

Fig. 10. Combination of structured BPMN processes and a flexible CMMN case embedded in a CMMN case model

Blaukopf and Mendling compare BPMN with CMMN against the background of organizational routines and their requirements. They find CMMN to better support characteristic routines, especially in regards to data sharing among tasks and participants [1].

Hewelt and Weske propose a hybrid approach in [9]. Modeled as small BPMN fragments, business scenarios are driven by *data objects* and their defined life cycles. They address the problem of arbitrary selection and cancelation of tasks in a structured BPMN model constrained by *sequence flows*. They argue that expressing the option to cancel an *activity* at different points throughout the process would necessitate many *gateways* and eventually lead to an unmanageable model.

In [4], Bruno addresses the inability to express organizational units and roles by proposing a life-cycle model to connect roles to entities/*case files* and *case file items*. Based on the GSM approach and CMMN *stages*, it links data inputs to *stages* and associates roles to them in order to model their data access using a UML class model structure and annotations on *tasks*. Different stereotypes are used to express a case manager role, primary entities and external as well as internal roles. *Stages* are used to group interrelated *tasks* together. *Stages* and *tasks* are annotated with attributes laid out in addition to the UML classes. The approach seems also applicable to BPMN *ad hoc sub-processes*.

Shahrah and Al-Mashari use CMMN to model emergency response processes in [21]. CMMN is used to model a suitable outline while retaining flexibility. In a concrete flood emergency scenario, *sentries* are used on *stages* to decide when interrelated tasks become necessary. A focal point in is a *repeatable decision task* used to trigger necessary actions. Execution semantics of the model are briefly discussed. Compared to

a previous approach to model the scenario using state charts, the CMMN model is considered to be advantageous.

Herzberg et al use CMMN to model variants in clinical pathways of hospitals in [8]. Processes previously modeled in BPMN are modeled using CMMN. In order to create a unified model which is valid across the surveyed hospitals, they use *discretionary items*, expressing differences and optionality.

VIII. Conclusion

We compare BPMN and CMMN models expressing a flexible process part employed for a specific component family, *relays*, to release components for production. The task list work requires a high degree of flexibility in execution and is knowledge-intensive and data-driven.

Using four general categories, we highlight the advantages and disadvantages of the flexible structures in BPMN (*ad hoc sub-process*) and CMMN (regular *case*), before deriving simple guidelines to support the decision of which notation to use for similar processes.

While offering some degree of flexibility with the *ad hoc sub-process*, BPMN is still bound to the overall imperative modeling, making arbitrary jumps difficult to model while providing some guidance to the workers. Data remains a second-class citizen and is usually restricted to single tasks, not to be shared and viewed by all process participants. The execution semantics of the *ad hoc sub-process* are limited. Two-way communications can be modeled in BPMN.

CMMN offers a better degree of flexibility and elements to express the work on the task list. Work can be grouped together in *stages* which have execution semantics. Optional work can be highlighted and sequence flows capturing all possible orders and (sub-)sets of *tasks* do not have to be modeled explicitly. The implicit routings and usage of *sentry criteria* may require a deeper domain knowledge than the same process expressed as a BPMN model might require. CMMN elements also support a goal driven, declarative approach to modeling.

One notable disadvantage is the lack of outgoing communication events, requiring either implicit communications or an embedded process task.

Finally, a combination of both is suggested. While not specified, CMMN models may be included in BPMN with the *call activity*. BPMN models can be embedded in CMMN with the *process task*. Used together, both highly structured and very flexible processes can be modeled.

References

[1] S. Blaukopf and J. Mendling. An organizational routines perspective on process requirements. In E. Teniente and M. Weidlich, editors, *Business Process Management Workshops*, pages 617–622, Cham, 2018. Springer International Publishing.

[2] Business process model and notation (BPMN) specification, 2011. Version 2.0.

[3] R. Breitenmoser and T. Keller. Case management model and notation - a showcase. *European Scientific Journal*, Vol.11, No.25, 09 2015.

[4] G. Bruno. Extending cmmn with entity life cycles. *Procedia Computer Science*, 121:98 – 105, 2017.

[5] C. Di Ciccio, A. Marrella, and A. Russo. Knowledge-intensive processes: Characteristics, requirements and analysis of contemporary approaches. 4:29–57, 03 2015.

[6] Case model management and notation (CMMN) specification, 2016. Version 1.1.

[7] R. M. de Carvalho, H. Mili, A. Boubaker, J. Gonzalez-Huerta, and S. Ringuette. On the analysis of cmmn expressiveness: Revisiting workflow patterns. In *2016 IEEE 20th International Enterprise Distributed Object Computing Workshop (EDOCW)*, pages 1–8, Sept 2016.

[8] N. Herzberg, K. Kirchner, and M. Weske. Modeling and monitoring variability in hospital treatments: A scenario using cmmn. In F. Fournier and J. Mendling, editors, *Business Process Management Workshops*, pages 3–15, Cham, 2015. Springer International Publishing.

[9] M. Hewelt and M. Weske. A hybrid approach for flexible case modeling and execution. In M. La Rosa, P. Loos, and O. Pastor, editors, *Business Process Management Forum*, pages 38–54, Cham, 2016. Springer International Publishing.

[10] R. Hull, E. Damaggio, F. Fournier, M. Gupta, F. Heath, S. Hobson, M. Linehan, S. Maradugu, A. Nigam, P.Sukaviriya, and R. Vaculin. Introducing the guard-stage-milestone approach for specifying business entity lifecycles. In M. Bravetti and T. Bultan, editors, *Web Services and Formal Methods*, pages 1–24, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[11] K.E. Kurbel. *Enterprise Resource Planning and Supply Chain Management: Functions, Business Processes and Software for Manufacturing Companies*. Progress in IS. Springer Berlin Heidelberg, 2013.

[12] M. Kurz, W. Schmidt, A. Fleischmann, and M. Lederer. Leveraging cmmn for acm: Examining the applicability of a new omg standard for adaptive case management. In *Proceedings of the 7th International Conference on Subject-Oriented Business Process Management*, S-BPM ONE '15, pages 4:1–4:9, New York, NY, USA, 2015. ACM.

[13] M. Marin and J.A. Brown. Implementing a case management modeling and notation (cmmn) system using a content management interoperability services (cmis) compliant repository. 04 2015.

[14] M. Marin, R. Hull, and R. Vaculín. Data centric bpm and the emerging case management standard: A short survey. In M. La Rosa and P. Soffer, editors, *Business Process Management Workshops*, pages 24–30, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[15] M. A. Marin, M. Hauder, and F. Matthes. Case management: An evaluation of existing approaches for knowledge-intensive processes. In M. Reichert and H.A. Reijers, editors, *Business Process Management Workshops*, pages 5–16, Cham, 2016. Springer International Publishing.

[16] M. A. Marin, H. Lotriet, and J. A. Van Der Poll. Measuring method complexity of the case management modeling and notation (cmmn). In *Proceedings of the Southern African Institute for Computer Scientist and Information Technologists Annual Conference 2014 on SAICSIT 2014 Empowered by Technology*, SAICSIT '14, pages 209:209–209:216, New York, NY, USA, 2014. ACM.

[17] A. Meyer, N. Herzberg, F. Puhlmann, and M. Weske. Implementation framework for production case management: Modeling and execution. In *Enterprise Distributed Object Computing Conference (EDOC), 2014 IEEE 18th International*, pages 190–199. IEEE, 2014.

[18] A. Meyer and M. Weske. Activity-centric and artifact-centric process model roundtrip. In N. Lohmann, M. Song, and P. Wohed, editors, *Business Process Management Workshops*, pages 167–181, Cham, 2014. Springer International Publishing.

[19] H.R. Motahari-Nezhad and K.D. Swenson. Adaptive case management: Overview and research challenges. In *2013 IEEE 15th Conference on Business Informatics*, pages 264–269, July 2013.

[20] A.-W. Scheer, O. Thomas, and O. Adam. *Process Modeling using EventDriven Process Chains*, chapter 6, pages 119–145. Wiley-Blackwell, 2005.

[21] A. Y. Shahrah and M. A. Al-Mashari. Modelling emergency response process using case management model and notation. *IET Software*, 11(6):301–308, 2017.

[22] A. Shtub and R. Karni. *ERP: The dynamics of supply chain and process management: Second edition*. Springer US, 2010.

[23] K. D. Swenson. State of the Art In Case Management. Technical report, Fujitsu, 03 2013.

[24] K.D. Swenson and L. Fischer. *How Knowledge Workers Get Things Done: Real-World Adaptive Case Management*. BPM and Workflow Handbook Series. Future Strategies, 2012.

[25] W.M.P. van der Aalst. Making work flow: On the application of petri nets to business process management. In J. Esparza and C. Lakos, editors, *Application and Theory of Petri Nets 2002*, pages 1–22, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[26] W.M.P. van der Aalst and M. Weske. Case handling: A new paradigm for business process support. *Data Knowl. Eng.*, 53(2):129–162, May 2005.