Comparing ConDec to CMMN
Towards a Common Language for Flexible Processes

Renata M. de Carvalho, Hafedh Mili, Javier Gonzalez-Huerta, Anis Boubaker and Abderrahmane Leshob

LATECE Laboratory, Université du Québec à Montréal, Montréal, Canada

Keywords: Business Process, Flexible Process, Case Management, Declarative Process, CMMN, ConDec.

Abstract: Flexible processes emerged to provide flexibility to business process execution. A flexible process is not static and can have several different executions, that is influenced by the current situation. In this context, the decision-making is placed in the hands of any knowledge worker during the execution, who decides which tasks and in which order they will be executed. Two approaches for flexible processes are discussed in this paper: case management and declarative processes. In particular we use the CMMN standard and the ConDec language for the two approaches, respectively. We compare them based on scope, model representation, formal semantics, and limitations. Our goal is to present commonalities and differences between the languages in order to identify potential extensions to make them more complete to attain more flexible process examples.

1 INTRODUCTION

Several approaches have emerged to provide more flexibility to business process execution (Nurcan, 2008). They argue that the number of unforeseen situations (exceptions) in a workflow model may not represent the alignment between process and real environment. Imperative models are either too simple, thus unable to handle the variety of situations that may occur; or too complex, trying to model every possible situation but being hard to maintain.

In order to cope with imperative processes shortcomings, authors have suggested different approaches to model business processes that we will call collectively *flexible processes*. In a flexible process, the stakeholders have more freedom to decide which sequence of tasks to perform in order to accomplish a given goal. In the paper, we discuss two different ways of supporting flexible processes.

Case Management (CM) falls in the first category by providing capabilities to customize processes at runtime (Motahari-Nezhad and Swenson, 2013). It adheres to the principle of planning-by-doing, considering the work context, and the ability to accommodate changes in the environment(Motahari-Nezhad and Swenson, 2013). This allows the stakeholders to respond to changes or unforeseen requirements that were not considered during designing the business processes (Kurz, 2013).

The declarative process falls in the second category providing flexibility by defining *what* must be executed without describing *how* the execution must take place (Pesic and van der Aalst, 2006). The model comprises only the definition of tasks and constraints that must be respected during the whole execution, abstracting away from any control flow and explicit sequencing considerations. Hence, any sequence of tasks can be executed as long as no constraint is violated. It is important to emphasize that in case management the constraints are also defined declaratively.

In this paper, we provide a detailed comparison of these two approaches for modeling business processes. In particular, we use the CMMN (Case Management Model and Notation) standard (for case management) and the ConDec language (for declarative processes) to evaluate a variety of similarities and differences throughout the paper. Moreover, we summarize the general background of each concept of flexibility and their originating purpose, we compare the two languages with respect to several dimensions (scope, model representation, and formal semantics), and we present some limitations to both languages.

This paper is structured as follows: Section 2 discusses more about flexibility in business processes; an overview of the CMMN standard is presented in Section 3; Section 4 presents the background of the ConDec language; the discussion and comparison between CMMN and ConDec is presented in Section 5;

Carvalho, R., Mili, H., Gonzalez-Huerta, J., Boubaker, A. and Leshob, A.
Comparing ConDec to CMMN - Towards a Common Language for Flexible Processes.
DOI: 10.5220/0005688002330240
In *Proceedings of the 4th International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2016)*, pages 233-240
ISBN: 978-989-758-168-7

Section 6 presents some related work; and final remarks and our conclusion in Section 7.

2 FLEXIBLE BUSINESS PROCESSES

In the flexible business process area, the word *flexible* means the business process is not static, it can change or get adjusted during its execution according to different situations. The goal is to be able to define flexible processes in such a way that allows users to handle unexpected situations safely and with adequate information during the process execution.

This quality, termed *flexibility*, reflects its ability to deal with changes, by varying or adapting certain parts of the business process that are affected by the changes, whilst retaining the essential format of those parts that are not impacted by the variations. Indeed, as has been noted, flexibility is as much about what should stay the same in a process as what should be allowed to change (Mulyar et al., 2007).

The idea of a flexible business process model is to give more flexibility to the process execution. The lack of flexibility in workflow systems has been already discussed by van der Aalst (van der Aalst et al., 2005). He pointed the following problems related to the fundamental concepts of the workflow approach:

- The transformation of non-atomic tasks (for real users) into atomic ones (for workflow system) in order to distribute work, even if the same work is actually being done at a more fine-grained level.
- The distribution and authorization problem, which coincides in contemporary workflow management systems. In real problems, work-items assigned to a worker does not coincide with work-items he may be authorized to do.
- Since workflow systems are focused on control flow, the context (data associated to the process and not just to tasks) does not receive enough importance and, sometimes, there is no support for defining it, resulting in errors and inefficiencies.
- The workflow approach is rigid and inflexible because it focuses on what *should* be done instead of what *can* be done.

In general, the knowledge user working on a process instance and executing a flexible process has the option to decide between several tasks that are enabled. However, with such a number of options, the user needs to have an in-depth knowledge about the process he is working on.

3 AN OVERVIEW OF CMMN

Case Management makes no distinction between design time and runtime. As explained in (Baresi and Ghezzi, 2010), it does not put all the process design responsibilities in the hands of a business analyst, who models and tests the process before it is executed. For a case management model, any knowledge worker is allowed to design and execute a process instance, called a *case* in case management.

Even if a business analyst provides templates for the process, a case can be customized during runtime by a case manager to fit his needs. The case manager is any knowledge worker that actually uses the process definition to execute a process instance. For example, for a health care process a doctor or a nurse can be case managers; for a police investigation, the case manager can be a detective.

Like any other flexible process, it is not possible to predict a case. The decision-making is placed in the hands of a case manager, who can directly modify the process. He can handle the case in any way that is necessary, as long as it respects the constraints imposed. This allows the case manager to optimize the process to the current case, improving the solution to a specific situation.

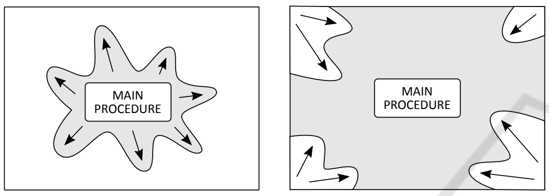
Case management grew based on two main concepts: i) context management, which addresses the need in many companies to maintain a great amount of documents and files; and ii) the management of cases. The Object Management Group (OMG) and the case management community worked to provide an industry wide standard called Case Management Model and Notation (CMMN) ((OMG), 2014).

For a CMMN model, there exist two phases. During the design phase, the business analyst defines predefined segments in the case model, and can also define discretionary items, which enable the case manager to customize the case during runtime. In the second phase, the case manager executes the process following the pre-defined plan. He can also modify the process design, instantiating the discretionary items by picking concrete ones depending on his needs.

4 AN OVERVIEW OF ConDec

ConDec is a *declarative* language for describing business processes (Pesic and van der Aalst, 2006). Its authors argue that, although adaptive workflow systems allow changes in an *imperative* process definition during runtime, such systems remain *imperative*. Although case management or other adaptive workflow systems offer the possibility to open, skip, execute and re-execute tasks, they still restrict the process execution (Pesic and van der Aalst, 2006).

The idea of declarative models is to change the nature of describing a process. The other modeling languages use an inside-to-outside approach, focusing on specifying first the main procedure, and then the possible deviations for the process (see Figure 1(a)). ConDec uses an outside-to-inside approach, focusing on specifying only the restrictions of the process (Pesic and van der Aalst, 2006) (see Figure 1(b)). In ConDec, the business analyst is guided to define *what* should be executed without specifying *how*. He is driven to define tasks presented in the process, and the relations between them. Each relation represents a constraint, which can be viewed as a policy or a business rule but does not describe an execution order.



(a) Inside-to-outside. (b) Outside-to-inside.

Figure 1: Difference between approaches. Arrows represent what is defined in design time and in gray, what is reachable at runtime (Based on (Pesic and van der Aalst, 2006)).

It is during the execution that a user decides which task, among the eligible ones, will be executed and how. During the execution, any task that does not violate a constraint can be chosen to be executed. This is considered an optimistic approach where, in principle, anything is possible unless it is explicitly forbidden. This leaves a lot of freedom for users, who can make decisions and work in various ways with the same ConDec model (Pesic and van der Aalst, 2006).

Modeling a process in ConDec, the business analyst can range from very relaxed models to very restrictive models. The most relaxed model contains only tasks defined without any constraint. Hence, the user has total flexibility to (re-)execute tasks following his own preference. A restricted model can be viewed as a model with constraints defined in such a way that it behaves like an imperative process.

5 COMPARING CONDEC TO CMMN

CMMN and ConDec are languages for modeling different categories of flexible business processes. Both of them describe tasks and constraints that must be respected during the execution. Here, we compare these two modeling languages concerning different aspects in order to highlight concepts presented in one language, both, or none of them. We compare the languages based on: i) scope, ii) model representation, iii) formal semantics, and iv) limitations. Other work (Gluch et al., 2001) argued about and used the same set of parameters to compare languages.

5.1 Scope

Both modeling language approaches were defined to focus on different aspects of a business process.

Case Management was created to model flexible and knowledge intensive business processes (van der Aalst et al., 2005), which means that it was created for coordination of work that is not routine and predictable, and requires human judgment (Motahari-Nezhad and Swenson, 2013). In case management group the discussion is around the case manager. They argue that one of the problems in modeling a business process is that a business analyst does not always have necessary knowledge. It is necessary to provide a way for the case manager to customize a case for his specific needs at runtime.

CMMN is a standard for dealing with cases. At design time, business analysts are engaged in modeling. They are concerned with the specification of: i) tasks that are always part of some segment; ii) constraints to guide the possible actions to be taken; iii) some decisions to influence future actions based on new facts or events; and, where appropriate, iv) defining potential upcoming items to be planned in runtime. At runtime, the case manager executes the case. He must perform tasks as planned, following constraints already defined. But the case may evolve by planning new tasks when possible.

The main particularity for the CMMN is the definition of potential upcoming items, which are called *discretionary* items. When a business analyst defines a discretionary task during the design time phase, he allows the case manager to add and plan, in this place, a specific task driving the execution. The CMMN standard also provides a *PlanningTable* to define the scopes where the case manager may operate.

Declarative approach was created to support dynamic processes in an important issue in rapidly changing organizations, whose agility is at a competitive level and subjected to frequent changes (Pesic and van der Aalst, 2006). In declarative group, the discussion focuses on restricting only the necessary, making the user (the worker who will execute the process) responsible for choosing the order of tasks at runtime.

ConDec is a language for defining declarative processes. At design time, the business analyst determines all possible tasks and the constraints to restrict things that shall not happen in the context. At runtime, the user is informed only about the enabled tasks at that moment and chooses, among these tasks, the next one to be executed.

The main particularity for ConDec is that the constraints represent dependencies/relations between tasks. Most of these relations do not have an immediate nature. For example, a *co-existence* relation indicates that if a certain task is ever executed, another task must also be executed eventually, and vice-versa. However, it does not say when the other (related) task must be executed.

5.2 Model Representation

In this section we compare the expressiveness of the two modeling languages through some important aspects. We will use an incremental example throughout this section to illustrate the differences between CMMN and ConDec. Using this example we are also able to illustrate different representations of similar concepts in the languages. The example, from the health care area, consists of a very simple representation of the process of an emergency department of a hospital. It summarizes the constraints that affect to the process that follows the arrival of a patient.

5.2.1 The Basics

The first concern in this process states that every patient that arrives at the emergency department should be assigned a triage acuity level. Hospital services, tests, procedures and interventions are also part of an emergency. In our example, we are only considering two kinds of tests: blood tests and x-ray tests. Neither test puts patients at risk, and thus, both can also be prescribed by nurses. Thereby, whenever a patient arrives at emergency, a nurse will assign a triage level. After that, the patient can be seen by a physician. Medical tests can be prescribed to be done before or after the physician examination.

For any business process modeling language, a task is the smallest unit of work. Tasks can be executed manually or automatically. Figure 2 shows a CMMN representation of the described emergency example, which presents four tasks: *Triage*, *Physician Care*, *Blood Test*, and *X-Ray*.

In this process, all the tasks are manual (illustrated with the person icon). In CMMN, a manual task can be *blocking* or not. A blocking manual task only allows the case to continue after the completion of its execution. A non-blocking manual task allows the case to continue at the beginning of its execution.

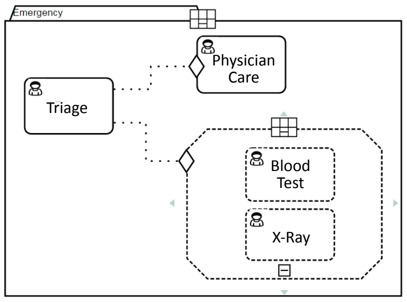


Figure 2: Emergency example defined in CMMN.

CMMN provides a way of grouping tasks through the *Stage* concept. *Stages* can be used to determine context inside the case. Figure 2 contains a Stage (the rectangular shape with angled corners) for representing the tests context available in the hospital. The Stage groups the two tasks related to the tests context.

The hollow diamond shapes in the borders of *Physician Care* task and the Stage are *Entry Criterion Sentries*. They are used in CMMN to define conditions to execute a task by defining an expression, which has two parts. The *IfPart* is used to define expressions based on variables, and the *OnPart* to relate the condition to events. The expression can be defined by one or both parts. When connected to a task (e.g. see dotted line between *Triage* and *Physician Care*), the sentry adds the task completion event to the sentry's *OnPart*. In our example, neither *Physician Care* nor the tests stage could be executed before *Triage* has been completed. One example of usage of the *IfPart* is for expressing that the patient should be seen by a physician only if the patient is assigned to triage level one (the highest priority for the triage systems). In this example, we would use both *IfPart* and *OnPart*. Note that the example using the *IfPart* would not allow the patient to be visited by a physician if it was assigned to another triage level.

The Stage, *X-Ray*, and *Blood Test* have dashed shapes to indicate that they are *discretionary* items. Every time a discretionary item is related to a context, the *Planning Table* icon appears in the context. The *Planning Table* allows the definition of *Applicability Rules*. They are used to determine whether an item is applicable, e.g. to determine for which roles a task is applicable. Hence, the discretionary items will be available to the case, when applicable, and may be executed depending on the case manager discretion.

The same process is defined using the ConDec language in Figure 3, which has the same tasks as in CMMN. The first thing to notice in Figure 3 is that ConDec does not provide a way for grouping tasks. And thus, *Blood test* and *X-Ray* cannot be grouped in a "tests context". Another thing is that ConDec does

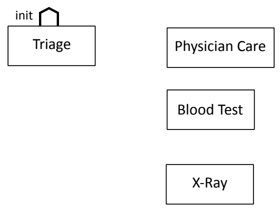


Figure 3: Emergency example defined in ConDec.

not have an equivalent to *discretionary* tasks.

To express the fact that *Triage* must be executed before any other task, we used the *init* constraint provided by ConDec, which means that the *Triage* task must be the first one to be executed. We could also have used the *precedence* constraint, which relates two tasks indicating that one must be executed before the other one. We could have expressed the same behavior by specifying three *precedence* constraints: from the *Triage* task to each of the other tasks.

Whenever a task is executed in CMMN, it cannot be re-executed. The same is true for the tasks inside the Stage: once a task is performed followed by a task from outside the Stage, it is not possible to go back to Stage and execute another task. Hence, the patient cannot pass through an x-ray, be seen by a physician, and then be required to do some blood test. To allow a behavior (as the re-execution of tasks) in CMMN, it must be specified. In ConDec, any of the tasks can be re-executed because there is no explicitly constraint prohibiting this behavior. Any execution trace containing the *Triage* task as the first one is valid. To prohibit a behavior in ConDec, it must be specified.

5.2.2 Prohibition, Counting and Immediate Behavior

The purpose of a triage system is to prioritize patients and identify those that cannot wait to be seen by a physician. Patients assigned to the highest priority level must be seen by the physician immediately. And every patient, regardless of their acuity level, must be seen by a physician before being discharged. One of the interventions that a physician can perform is the prescription of drugs. Some drugs can interfere with blood tests, changing results. For the sake of simplicity, we will not make a distinction between drugs. We will assume that any drug can alter any blood test, and hence, will forbid blood tests on patients who have taken drugs. Another limitation regarding the tests: x-ray tests can only be performed twice on the same patient, due to the radiations exposure.

Figure 4 depicts the representation of the emergency example with the additional constraints in CMMN. This model has a new task called *Take Prescribed Drugs*. This task is associated with an *Entry Criterion Sentry*, stating that the patient can take prescribed drugs only after being seen by a physician.

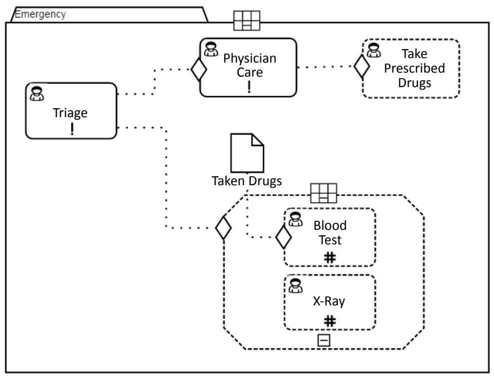


Figure 4: Emergency example defined in CMMN.

In CMMN, and case management, one of the concepts that guides a case is its context, represented in a *CaseFile*. In Figure 4, the *CaseFile* has the information about drugs taken. In this example, drugs can be taken after a *Physician Care*, if prescribed, or they can have been taken before the patient arrives at the emergency (if informed by the patient). Thus, *Blood Test* will only be available if there is no information about drugs taken in the *CaseFile*.

CMMN uses decorators to indicate particular behavior patterns. Each decorator specifies a different pattern and can be applied to a different set of items. The criterion sentries are examples of decorators. Moreover, in Figure 4 we use two other decorators: *Required*, denoted by !; and *Repetition*, denoted by #. The *Required* decorators in *Triage* and *Physician Care* tasks indicate that these are mandatory tasks and the process cannot finish without executing them. The *Repetition* ones in *Blood Test* and *X-Ray* tasks imply that these tasks can be re-executed. We could also add a *Repetition* decorator to the *Stage* to indicate that the tests context can also be visited more than once, e.g., allowing tests to be performed before and after a physician care.

Rules will be used to express other concerns. For expressing that patients with the highest priority triage level must immediately be seen by a physician we will use an *Applicability Rule*. This rule certifies that the Stage will only be applicable for patients with different triage level after the *Physician Care*. The limited amount of x-rays will be expressed by using a *Repetition Rule* to restrict the use of the repetition decorator in the *X-Ray* task.

The ConDec model with the same concerns is shown in Figure 5. The new task, *Take Prescribed Drugs*, can only be executed after a physician care. This relation is defined by a *precedence* constraint indicating that the execution of *Physician Care* must precede the execution of *Take Prescribed Drugs*.

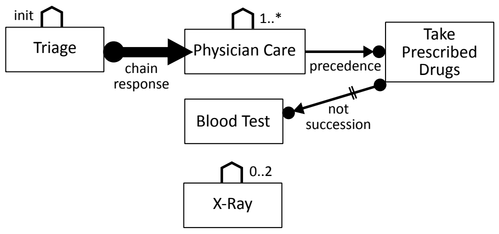


Figure 5: Emergency example defined in ConDec.

Four other constraints are new in the ConDec model. The *not succession* constraint indicates that after taking prescribed drugs, it is not possible to require blood tests for the patient. We could use a global variable, known by the whole process, to have a representation as the one used in the CMMN's case file. The *existence* template allows the business analyst to define a minimum amount of executions for a task. In Figure 5 the *existence* constraint is defined to the *Physician Care* task, which must be executed at least once. The third constraint limits the maximum number of times an x-ray can be performed by using an *absence* constraint. For this example, it was possible to express that the x-ray test can be prescribed at most twice. The fourth constraint, the *chain response* constraint, is related to the patients assigned to the highest acuity level during the triage. All *chain* templates in ConDec have an immediate behavior. Thus, if the patient is associated to the highest acuity level (limited by an expression not visible), the *Physician Care* must be executed immediately after the *Triage*, disabling all the other possibilities.

5.2.3 Events

Each hospital has its own policy and arrangements for discharging patients. Generally, whenever the patient is assessed and is considered medically fit, or his treatment is no more urgent, he/she can receive a discharge authorization. The discharge of a patient characterizes the end of the patient's process execution inside the hospital. In addition to the discharge authorization, the patient can also receive a care plan to be followed outside the hospital.

The CMMN model presented in Figure 6 represents the *Discharge Authorization* as an event (represented by a double circle). An event is any action or occurrence detected at any time that can be handled by the model. In CMMN, an event may trigger the enabling, activation, and termination of another item.

An *Exit Criterion Sentry* is similar to the *Entry Criterion Sentry* and can be expressed by an *IfPart* (for variables), an *OnPart* (for events), or both. The

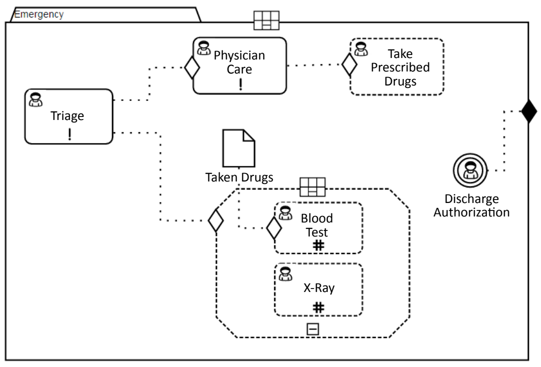


Figure 6: Emergency example defined in CMMN.

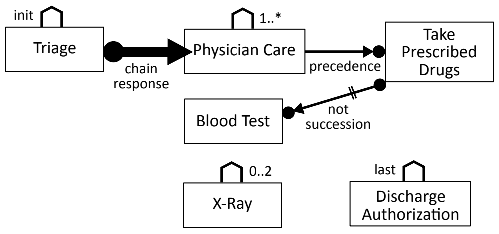


Figure 7: Emergency example defined in ConDec.

Exit Criterion Sentry is represented by a solid diamond shape. When the conditions of an *Exit Criterion Sentry* are true, the related item will be terminated (exited). In our example, an *Exit Criterion Sentry* is used directly to the *Case Plan Model*. We used an *UserEventListener*, which is raised by a user, for terminating the case execution.

ConDec does not allow the specification of events. The business analyst may define *Discharge Authorization* as a task, as in Figure 7.

The *last* constraint is similar to the *init* one, defining the position of task during the execution. In this case, the related task (*Discharge Authorization*) must be the last one executed.

5.3 Formal Semantics

All modeling languages must have a well-defined underlying semantics, which helps to understand the defined process and its execution.

The CMMN standard is a recent OMG standard. In 2010 OMG published a request for proposals for a Case Management Process Model standard (Marin et al., 2013), and since then CMMN has been developed. Its first version was launched in 2014 and does not include a formal semantics. Also, there is no published work proposing a formalism for CMMN yet.

Regarding the ConDec language, it uses Linear Temporal Logic (LTL) as its formalism for the internal representation of business processes. Each template is mapped to an LTL formula, so that a Con-

Dec model can be interpreted in terms of LTL in a finite trace (Maggi et al., 2012). Due to this formalization, a runtime verification can be done focusing on violations from the interference of multiple constraints. Process enactment requires the construction of a Büchi automaton that contains all possible states of the process, which provides diagnosis about business constraints during the execution of the model.

This strategy is not very efficient, since it grows exponentially in size for realistic models. To solve this, REFlex (De Carvalho et al., 2013), another technique for modeling declarative business processes on top of ConDec, proposed a graph-based mechanism that does not share the same inefficiency. The graph represents only the current state of execution, which is updated at runtime. Through the graph, REFlex can infer future states and avoid inconsistent states. REFlex also made a different formalization for the ConDec templates using the Alloy language (Jackson, 2006), which ensures that the graph used is a precise representation of the model and also ensures that business processes will be correctly executed.

5.4 Limitations

We have already discussed some of the limitations of the two modeling languages in the previous sections. Here we discuss other limitations that are common to both CMMN and ConDec.

Other imperative modeling languages such as EPC and BPMN support to express resource allocation on a high-level model. On the executable level, languages like BPEL are able to express resources. A resource may be a human being or a machine, a service, a material. Having resources modeled in a process allows the definition and analysis of competition concepts, based on resources limited availability or consumption. Neither CMMN nor ConDec provides intrinsic definition of resource, driving the business analyst to make a simplified model of the reality without the resources definition, or to use other artifacts (e.g., addressing the resources competition manually) to deal with the involved resources.

Temporal constraints is another aspect that got the attention of other researchers, for example, to extend BPMN with such characteristics (Gagne and Trudel, 2009). The temporal notion presented in CMMN is only when using a *TimerEventListener*, to catch predefined elapses of time that trigger other actions. In ConDec, the notion of time is more scarce. There is only the notion of the immediate templates. For all other templates, the dependency must occur, but at any time in the future. Other temporal notions can be used in expressions for both languages but they should be interpreted by the execution engine.

6 RELATED WORK

In this section we present other works that also compared business process modeling languages. All of them compare imperative languages and/or languages that are not specific for business processes.

Milton and Johnson (Milton and Johnson, 2012) made a comparison between service blueprint and BPMN. They analyzed which service blueprint concepts are supported, partially supported or cannot be expressed in BPMN. After the analysis they found out that BPMN can be used to diagram service process, but in a different perspective compared with service blueprint. They are also confident that BPMN extensions can help improving the understandability of critical touch points driving service satisfaction.

Recker (Recker, 2010) compared two different languages that are specific for modeling imperative business processes: BPMN and EPC. His comparison focused on the grammars of each language, showing differences in continued usage behavior of the two styles of grammar. He also conducted a study to determine significant factors that impact in language usage, suggesting that the ease of use perceptions have a strong impact to user satisfaction and willingness to continue to use a process modeling grammar.

Another work made by Marin, Lotriet, and Van Der Poll (Marin et al., 2014) makes the comparison between business process languages. For this work, CMMN is also part of the study. The authors did the exploratory research to understand CMMN's complexity. For this research, they compared CMMN with BPMN, EPC, and UML AD in terms of the quantity of objects, relationships and properties for modeling the same problem. They used the cumulative complexity to determine the difference between the languages, showing that CMMN is favorable in comparison to BPMN, which makes it much promise.

7 DISCUSSION

This paper presented two different modeling languages for flexible processes: CMMN and ConDec. Even though we know that both languages have different purposes, and so they are based on different concepts, both have the main goal of providing flexibility to business process specification and execution.

CMMN is an OMG standard for case management. The concern is that cases have their particularities and cannot be anticipated at design time. Furthermore, the case manager should be able to customize it depending on his needs at runtime. CMMN provides the representation of discretionary items, which may become concrete at runtime. Hence, CMMN is a planing-by-doing language, also considered as an inside-to-outside approach because the business analyst must represent mostly the possibilities of execution, even with some flexibilities.

In ConDec, the principle is that the execution is guided by constraints, and the stakeholder executing the process can choose the order of tasks at runtime. Any order is possible to be chosen since it does not violate any constraint. It is considered an outside-to-inside approach, since the business analyst must be worried about the constraints (the restrictions to the model) and not about the possible execution paths.

We compared these two languages based on other aspects. ConDec was developed in 2006 (Pesic and van der Aalst, 2006), already having formal semantics defined in LTL. Each ConDec template can be mapped to an LTL formula, and the resulting one is used for the enactment of process at runtime. There are other related works improving its execution semantics, e.g. (De Carvalho et al., 2013). CMMN is more recent, with its first version released in 2014. Because of that, few works can be find related to CMMN and it has no formal semantics yet.

In addition to present commonalities and differences between languages, we pointed that the definition of resources is not intrinsically addressed by both languages and the definition of time is really limited in both of them. All this comparison makes us believe that both languages can be extended in order to be more complete in the context that it would be possible to attain more flexible process examples.

ACKNOWLEDGEMENTS

This study is supported by NSERC (Natural Sciences and Engineering Research Council of Canada).

REFERENCES

Baresi, L. and Ghezzi, C. (2010). The disappearing boundary between development-time and run-time. In *Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research*, FoSER '10, pages 17–22, New York, NY, USA. ACM.

De Carvalho, R., Silva, N., Lima, R., and Cornelio, M. (2013). Reflex: An efficient graph-based rule engine to execute declarative processes. In *Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on*, pages 1379–1384.

Gagne, D. and Trudel, A. (2009). Time-bpmn. In *Commerce and Enterprise Computing, 2009. CEC '09. IEEE Conference on*, pages 361–367.

Gluch, D., Comella-Dorda, S., Hudak, J., Lewis, G., Walker, J., and Weinstock, C. (2001). Model-based verification: Scope, formalism, and perspective guidelines. Technical Report CMU/SEI-2001-TN-024, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA.

Jackson, D. (2006). *Software Abstractions: Logic, Language, and Analysis*. The MIT Press.

Kurz, M. (2013). Taming diversity: A distributed acm-based approach for cross-enterprise knowledge work. In *Web Intelligence (WI) and Intelligent Agent Technologies (IAT), 2013 IEEE/WIC/ACM International Joint Conferences on*, volume 3, pages 87–91.

Maggi, F. M., Westergaard, M., Montali, M., and van der Aalst, W. (2012). Runtime verification of ltl-based declarative process models. In *Runtime Verification*, volume 7186 of *LNCS*, pages 131–146. Springer Berlin Heidelberg.

Marin, M., Hull, R., and Vaculn, R. (2013). Data centric bpm and the emerging case management standard: A short survey. In *Business Process Management Workshops*, volume 132 of *LNBIP*, pages 24–30. Springer Berlin Heidelberg.

Marin, M. A., Lotriet, H., and Van Der Poll, J. A. (2014). 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*, pages 209:209–209:216, NY, USA.

Milton, S. K. and Johnson, L. W. (2012). Service blueprinting and bpmn: a comparison. *Managing Service Quality: An International Journal*, 22(6):606–621.

Motahari-Nezhad, H. and Swenson, K. (2013). Adaptive case management: Overview and research challenges. In *Business Informatics (CBI), 2013 IEEE 15th Conference on*, pages 264–269.

Mulyar, N. A., Schonenberg, M. H., Mans, and van der Aalst (2007). Towards a taxonomy of process flexibility. *BPM Center Report BPM-07-11*.

Nurcan, S. (2008). A survey on the flexibility requirements related to business processes and modeling artifacts. In *HICSS '08: Proceedings of the 41st Annual Hawaii International Conference on System Sciences*, page 378, Washington, DC, USA. IEEE Computer Society.

(OMG), O. M. G. (2014). Case management model and notation (cmmn) version 1.0.

Pesic, M. and van der Aalst, W. (2006). A declarative approach for flexible business processes management. In *Business Process Management Workshops*, volume 4103 of *LNCS*, pages 169–180. Springer Berlin Heidelberg.

Recker, J. (2010). Explaining usage of process modeling grammars: Comparing three theoretical models in the study of two grammars. *Information & Management*, 47(56):316 – 324.

van der Aalst, W. M., Weske, M., and Grnbauer, D. (2005). Case handling: a new paradigm for business process support. *Data & Knowledge Engineering*, 53(2):129 – 162.