Enhancing Declarative Process Models with DMN Decision Logic

Steven Mertens(✉), Frederik Gailly, and Geert Poels

Department of Business Informatics and Operations Management,
Faculty of Economics and Business Administration, Ghent University,
Tweekerkenstraat 2 9000, Ghent, Belgium
{steven.mertens,frederik.gailly,geert.poels}@ugent.be

Abstract. Modeling dynamic, human-centric, non-standardized and knowledge-intensive business processes with imperative process modeling approaches is very challenging. Declarative process modeling approaches are more appropriate for these processes, as they offer the run-time flexibility typically required in these cases. However, by means of a realistic healthcare process that falls in the aforementioned category, we demonstrate in this paper that current declarative approaches do not incorporate all the details needed. More specifically, they lack a way to model decision logic, which is important when attempting to fully capture these processes. We propose a new declarative language, Declare-R-DMN, which combines the declarative process modeling language Declare-R with the newly adopted OMG standard Decision Model and Notation. Aside from supporting the functionality of both languages, Declare-R-DMN also creates bridges between them. We will show that using this language results in process models that encapsulate much more knowledge, while still offering the same flexibility.

Keywords: Business process modeling · Declarative process models · Decision logic · Decision management · Healthcare processes

1 Introduction

BPMN takes an imperative approach to business process modeling as it provides a precise graph-based definition of the process control-flow [1]. While BPMN is suitable for modeling static and standardized business processes [2], specifying the complete control-flow for each variation of processes that require a high degree of run-time flexibility[1] is time consuming and results in overly complex models.

Goedertier et al. [1] state that dynamic, human-centric, non-standardized and knowledge-intensive business processes (KiP) are most likely to require the run-time flexibility offered by declarative process modeling. While imperative approaches focus on explicitly defining the exact path of activities to reach the process goals, declarative approaches determine only the activities that may be performed as well as constraints prohibiting undesired behavior [3]. Applying a declarative modeling

[1] Run-time flexibility: the flexibility allowed by a process after being deployed [22].

K. Gaaloul et al. (Eds.): BPMDS 2015 and EMMSAD 2015, LNBIP 214, pp. 151–165, 2015.
DOI: 10.1007/978-3-319-19237-6_10

approach results in the specification of a collection of rules, constraints and assumptions that leaves enough freedom for various execution paths towards the process goals to exist. Additionally, explicitly specifying rules that remain tacit with imperative modeling, can enhance the knowledge management capabilities of the organization, allow for reuse of the rules in other process models, improve maintainability by way of high design-time flexibility[2], increase process compliance and improve traceability [4].

One of the most popular declarative process modeling languages is Declare [5, 6] (previously known as ConDec[3]). This language is based on Linear Temporal Logic (LTL), which is a formal language to express statements in modal temporal logic. It is very well suited to represent the control-flow of a process in a declarative manner as it does not offer a precise specification of the control-flow, but rather marks the rules to which a valid control-flow must oblige. Declare offers visual constructs to hide some of the complexity of LTL rules. To further improve its expressibility an extension has been proposed, Declare-R [7], that adds a resource perspective to the language.

The healthcare sector is one of the sectors where process modeling has had a hard time manifesting itself, as it remains mostly data-driven due to its knowledge-intensive nature. However, some of the main concerns trending in eHealth are very similar to other sectors, namely cost reduction and efficiency [8]. So a traditional focus on these two process goals could still create considerable value for both the patients as the healthcare personnel. The fact that process-orientation is nearly absent can be attributed mainly to the need to deliver a flexible and dynamic service. Medical professionals need to be prepared to handle a vast array of cases, where doctors are empowered to use their knowledge and judgment as a guide through the critical data-intensive situations. This makes it hard for traditional process modeling techniques to create added value and calls for a different approach. Languages like GLIF [9], Asbru [10] and PROforma [11] have been proposed to model the guidelines, used throughout the healthcare sector, visually as what are called Computer-Interpretable Guidelines (CIG). Mulyar et al. [12] state that these languages have problems with the dynamic and flexible nature of the healthcare processes (i.e., because they are hybrids of imperative modeling languages with decision modeling languages) and advise the use of a declarative language, CIGDec, which has since been integrated into Declare.

In this paper we will put Declare-R to the test by using it to model a realistic flexible process using a case example from the emergency department of a hospital. This will demonstrate that languages like Declare-R and CIGDec can model these types of processes well, but that they are missing essential information. This information can be seen as the intelligence of the process: the decision logic. The decision logic determines when a certain activity should be executed and this goes further than just specifying sequence constraints between activities. In imperative modeling languages like BPMN decision rules are implicitly modeled as part of the process control-flow (e.g., split gateways). Recently, OMG adopted (currently in finalization phase) a modeling method that allows for the separation of process logic and decision logic:

[2] Build-time or design-time flexibility: the intrinsic flexibility of a created model [22].

[3] http://www.win.tue.nl/declare/2011/11/declare-renaming/

the Decision Model and Notation (DMN) [13]. By separating decisions from the process control-flow, the decision rules can be explicitly specified. This means that the decision logic can be reused, be adjusted to evolve within an ever-changing environment and be used as justification for the choices being made. The decision logic is also specified in a declarative way [14], which makes it very well suited, if not more, to complement declarative process languages too. Consequently, it allows us to improve the way we model KiPs and the way we manage these processes. This results in value creation for all the stakeholders. This in turn can be an important incentive to establish a more process-minded way of thinking in KiPs.

The goal of the paper is to demonstrate how a combined Declare-R and DMN approach, we will call it Declare-R-DMN, can model flexible processes more completely with respect to their control-flow and decision logic, while still allowing for run-time flexibility. We will do this by first showing what can and cannot be modeled with Declare-R starting from a realistic healthcare process. Next, we will see what tools DMN can provide us with to model decisions, deontic rules and preferences. Finally, we discuss how the combined approach creates additional value, when compared to the use of only declarative process modeling, as it adds the decision logic which is an essential part of KiPs.

This paper is structured as follows. Section 2 gives a quick overview of the modeling languages used in this paper. In section 3, a case is presented, that demonstrates a realistic, dynamic and flexible process. This case will be modeled using Declare-R and Declare-R-DMN in section 4. In section 5 we provide a brief analysis of Declare-R-DMN. Finally, we conclude the paper and describe the future work in section 6.

2 Background

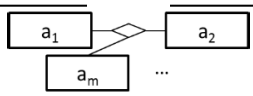
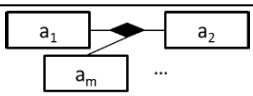
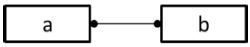
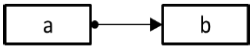
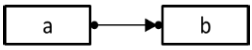
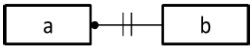
2.1 Declare-R

Declare is a graphical representation language proposed for declarative modeling of business processes based on LTL-logic [1, 5, 6]. A Declare model contains a set of activities and a set of constraints that can span multiple activities. It specifies the process environment in terms of what is necessary and what is not allowed (i.e., rules expressing the modal verb ‘must’), restricting the possible process executions. Contrary to other declarative languages, Declare supports optional constraints (i.e., using a dotted line instead of a solid line). Such constraints offer guidance (i.e., rules expressing the modal verbs ‘should’ and ‘ought to’) through knowledge-intensive activities [1], while their soft character ensures flexibility is maintained (i.e., it is not necessary to enforce them).

There are four groups of Declare constraints (see **Table 1**):

- Existence constraints: unary cardinality constraints predicating the number of possible executions of an activity.
- Choice constraints: n-ary constraints expressing a choice between activities.

Table 1. Constraint templates of Declare

Type	Name	Graphical	Meaning
Existence	absence(n+1, a)		Activity a can be executed at most n times
	existence(n, a)		Activity a must be executed at least n times
	exactly(n, a)		Activity a must be executed exactly n times
	init(a)		Activity a must be the first executed activity
	absence(n+1, a)		Activity a can be executed at most n times
Choice	choice(n of m, [a_1,…,a_m])		At least n distinct activities among $a_1,\ldots,a_m$ must be executed
	ex_choice(n of m, [a_1,…,a_m])		Exactly n distinct activities among $a_1,\ldots,a_m$ must be executed
Relation	resp_existence(a, b)		If *a* is executed, then *b* must be executed at any time before or after *a*
	coexistence(a, b)		Neither *a* nor *b* is executed, or they are both executed
	response(a, b)		If *a* is executed, then *b* has to be executed at any time after *a*
	precedence(a, b)		*b* can be executed only if *a* has been at some time previously executed
	succession(a, b)		*a* and *b* must be executed in succession (= response + precedence)
	…		
Negation	resp_absence(a, b)		If *a* is executed, then *b* can never be executed
	…		

- Relation constraints: binary constraints enforcing the presence of an activity in combination with another activity. Table 1 presents the five most important constraint templates. There are six additional templates based on two variations (i.e., alternate and chain) of the response, precedence and succession templates.

The relation templates can also be extended to involve more than two activities. See [6] for more details.

- Negation constraints: negative version of the relation constraints. This is graphically represented with two parallel lines perpendicularly crossing the representation of the relation constraint in question.

To improve the expressibility and practical usability, an extension was proposed, called Declare-R [7], which allows for a textual specification of the information needed to reason about resources:

- Estimates of the duration of each activity
- The available resources
- For each activity, the resource(s) required for its execution

2.2 DMN

The primary goal of Decision Model and Notation (DMN) [13] is to provide a common notation for decision logic that is understandable for business users, business analysts and technical developers. DMN provides the constructs to model decision rules and the decision-making process itself. A DMN decision model consists of two levels: the decision requirements graph (DRG) and the decision logic. The former describes where the required information is coming from and can be depicted in one or more decision requirements diagrams (DRDs). The latter describes the logic behind the decision, which is depicted in Decision Tables [15]. The upper half of a decision table specifies the possible combinations of conditions that lead to certain actions, while the bottom half contains the actions to be taken (i.e., outcomes). A minimal scope is specified for the standardization by OMG, but the goal is to offer support for other decision logic notations (e.g., decision trees) and allow for references to other types of models (e.g., SBVR).

3 Case Example

We elaborate further on a case from [5] with additional information provided by a practicing surgeon[4]. The process of treating arm-related fractures takes place in the emergency department of a hospital. The process entails the registration, diagnosis and treatment phases for patients with one or more fractures of a finger, hand, wrist, forearm, upper arm, shoulder, and/or collarbone.

The process starts when a patient is registered at the reception of the emergency department. Alternatively, in acute emergency situations, the registration can be done at a later time. The next step will usually be to examine the patient. During this examination, the doctor will make a list of the symptoms (e.g., excessive pain and deformation) of the patient. Based on these symptoms he will make a preliminary diagnosis. Normally, this diagnosis is checked by making X-rays, which in turn always results in

[4] Dr. Kjell Fierens, AZ Sint-Lucas in Ghent

a new examination to evaluate the X-rays and make the actual diagnosis. In some situations the doctor can make the final diagnosis without X-rays (e.g., clearly no fracture) or there is just no time for this due to emergency conditions.

The next phase involves the treatment the patient will receive. Of course, a treatment is only possible after at least a preliminary examination by a doctor. There are five types of treatment: bandaging, providing support with a sling, fixating the fracture, applying a cast or performing surgery. Each patient will receive at least one of these treatments, even when no fracture is present the patient will be bandaged or receive a sling. Choosing one does not eliminate other treatments, as some strategies combine two or more treatments and patients can be treated for multiple fractures simultaneously. While some treatments do not necessarily require follow-up activities, others might require physiotherapy. For example, muscle atrophy will quickly take its toll after applying a cast or after surgery (due to the usual postoperative period of rest). With the latter the additional damage done to the muscles should also be considered. The other treatments might require physiotherapy, but this is more case dependent. It is also possible that the patient receives a sling (of course no more than one for each arm) or some bandages at any time during the process in order to make he/she as comfortable as possible, no matter the diagnosis.

When we look at the case on a more detailed level we can identify several different variations. These represent the classes of fractures that can occur. Each has a specific flow and different characteristics to be taken into account. One common characteristic is that all fractures require surgery if the fracture is open or complex or when there is extensive damage to the arteries or nerves. Also, if there is no emergency situation, the diagnosis will need to be confirmed by an X-ray. If the patient has multiple fractures, one process instance can combine more than one of these variations.

- A fractured finger or a fractured bone in his hand: in most cases a simple fixation is enough to let it heal. The patient will receive a sling before being sent home.
- A fractured wrist: a cast will be applied, possibly after performing surgery. Surgery is required if the patient is a child (under 16 years old) and also has a damaged periosteum. For adults, surgery is only performed when dealing with open or complex fractures. Afterwards a follow-up X-ray will be taken to confirm that the bone is positioned correctly to start the healing process. The patient will receive a sling before being sent home.
- A fractured forearm: usually this requires no more than a cast. Only when the bone parts are too far apart, surgery is required. To support the cast, the patient receives a sling before being sent home.
- A fractured upper arm: is commonly treated by applying a fixation. If the fracture is an open, surgery is performed. This surgery is also performed when the patient has broken both arms, there is extensive artery damage, there is extensive nerve damage or when there is no improvement over a period of 3 months. The patient will receive a sling before being sent home.
- A fractured shoulder: usually the conservative treatment is enough, letting the shoulder heal while wearing a sling. In the other cases, surgery is required. Physiotherapy is also needed, because the shoulder joint will be inactive for an extended period during each of the two treatments.

- A fractured collarbone: is treated in most cases by resting it while wearing a figure of eight bandage. Surgery is only required when dealing with open or complex fractures or extensive damage to the arteries or nerves.

Additionally, if surgery is required for a broken wrist or forearm, but the OR is unavailable, a temporary cast will be applied to bridge the time until surgery.

Another aspect is the prescription of medication. There is a general policy that states that no medication can be prescribed without being proceeded by an actual doctor's examination. For pain medication it also requires the doctor or surgeon to agree that the patient is in pain or could be in pain in the nearby future. Furthermore, the policy makes a distinction between patients between 0-16 years old (we will refer to them as children) and the older patients (we will refer to them as adults). For instance, if a child had surgery or is in excessive pain as determined by a doctor's examination, he/she will always receive a prescription for a weak painkiller at first. Only if the doctor finds this to be insufficiently effective, a stronger painkiller can be prescribed. This also holds for adults, except after surgery, when stronger painkillers will be prescribed immediately. For both children and adults, after surgery they will be prescribed anticoagulants and anti-inflammatory drugs as precaution. Likewise, patients that received a cast could be prescribed anticoagulants. Because there exist strong painkillers that do not mix well with anticoagulants and anti-inflammatory drugs, a distinction is made between classes of strong painkillers:

- Strong painkillers A: should not be taken while on anticoagulants or anti-inflammatory drugs, but are preferred in other cases.
- Strong painkillers B: can be taken while on anticoagulants or anti-inflammatory drugs.

Furthermore, we also need to consider that some activities require the availability of certain resources, which in turn are limited in number. Also, they are not only used for this process, but rather represent a pool of resources shared among multiple independent processes of the hospital. The inventory of the resources and the activities that require then are as follows:

- 3 reception desks are used to register patients
- 15 exam rooms are used for the examination of the patient as well as for applying a cast, an external fixation, a sling and bandages
- 1 X-ray room with 1 X-ray machine used to make X-rays of the patients
- 4 operating rooms where surgery is performed on the patients
- 60 beds where the patients can rest after surgery
- 2 physiotherapy rooms used to provide in-house physiotherapy

Furthermore, there is also a list of human resources available:

- 3 receptionists to work at the reception
- 3 doctors to examine and treat patients (except for surgery)
- 10 nurses to apply casts/fixations/bandages or man the X-ray machine
- 2 surgeons to perform surgery
- 1 physiotherapist to provide physiotherapy sessions

Finally, estimates of the duration of the activities are provided. Of course, this is just an indication as this is dependent on the circumstances.

- Registration: 10 minutes
- Examination: 10 minutes
- Take an X-ray: 30 minutes
- Applying a cast: 15 minutes
- Applying a fixation: 10 minutes
- Applying a bandage: 5 minutes
- Applying a sling: 3 minutes
- Performing surgery: 120 minutes
- Resting after surgery: 180 minutes
- Physiotherapy: 60 minutes
- Prescribing painkillers, anticoagulants or anti-inflammatory drugs: 1 minute

4 Case Models

4.1 With Declare-R

If we model the process using Declare, we obtain the model in **Fig. 1** (the wavy line constraints are explained in section 4.2). By using the Declare-R extension of Declare we can also incorporate the resource constraints of the case as described in black in **Table 2**.

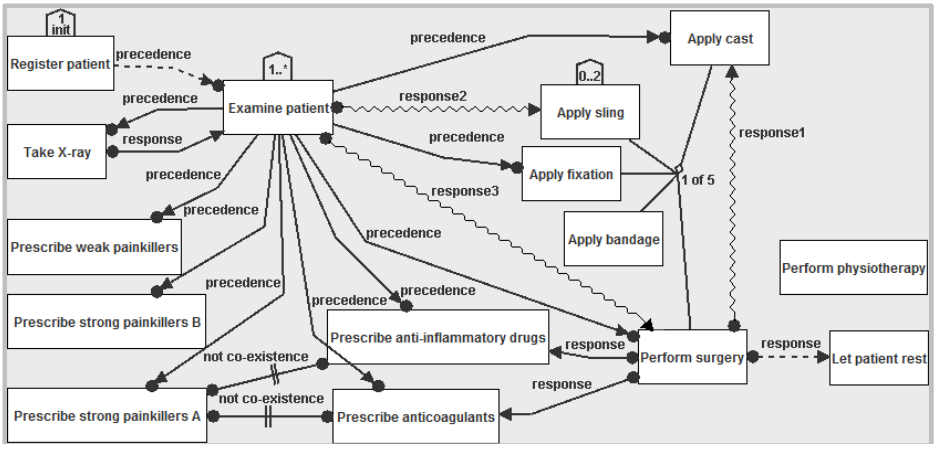


Fig. 1. Declare model of the arm fracture case

An important aspect of the case is the treatment of different types of fractures, each with a unique course of action. The Declare-R model describes a set of sequencing, timing and resource constraints, but does not consider this aspect. Creating a hierarchical process model is not supported according the original definition of Declare [6],

however Zugal et al. [16] have described and evaluated a way to introduce it. But even hierarchy would not be enough to model these variations. This is because instead of specifying variations on one general activity, each variation adds different existence and sequential constraints between the activities that are already present on the general level. Each variation could be modeled in a separate Declare model, but how do we know which model is applicable in which situation (i.e., need for run-time flexibility)?

Each variation is in essence a different diagnosis, which can occur simultaneously (e.g., fracture in wrist and forearm). Since this diagnosis is not available at the start of the process, the type of treatment is chosen at runtime, and thus it is a decision made by the doctor conducting the examination. This decision does not only require information of the previously executed steps, but also information about the patient, the symptoms, the test results and perhaps the resource availability. Declare(-R) lacks expressibility to model these decisions and the data on which they are based.

Another shortcoming of this model is the absence of role responsibilities, which could lead to misuse of the model. Not every process actor can initiate each activity (e.g., a nurse cannot perform surgery). Support is however added to the official Declare tool, so this is a minor issue.

4.2 With Declare-R-DMN

The Declare-R-DMN language that we propose incorporates the Declare-R model presented in Fig. 1 and Table 2, while including the decision logic that was missing. The role responsibilities have also been explicitly added (in gray) in Table 2, specifying the process actors that can execute a certain activity.

By using DMN, as part of the proposed Declare-R-DMN language, to model the decision concerning the appropriate treatment, we get the Decision Requirements Graph (DRG) from Fig. 2 and the decision logic (i.e., decision table) from Table 3.

The decision requirements graph in Fig. 2 visualizes where we get the information needed to make the decision concerning the treatment to be applied. The patient record, examination notes and X-ray are documents containing explicit knowledge. This is combined with the implicit knowledge and experience of the responsible doctor to reach a decision on what treatment is appropriate.

The decision logic in Table 3 is presented in a simple syntactic and standardized structure [15], modeling the cause-and-effect relationships between the conditions and actions. The activity corresponding to the action of a decision table has to be executed at any time in the future. This means that other activities could be executed before the action activity, but the action activity has to be executed eventually. This is similar to how the response-constraint template of Declare works (see Table 1). For example, if the patient has an open (visually determined) or complex (determined with X-ray) fracture of the finger, surgery will be performed, but another X-ray might be taken first. The alternatives are visualized side by side to facilitate the analysis of combinations. The completeness property guarantees that every combination of condition values is considered [14]. Because of this structure the decision conditions are easy to understand and manipulate by analysts, programmers and non-technical users [15]. This makes it a great medium for documenting decisions and to allow for backwards

Table 2. Declare-R resource constraints of the arm fracture case

Estimates	**Resource requirements and role responsibilities**
Duration(Register patient) = 10 Duration(Examine patient) = 20 Duration(Take X-ray) = 30 Duration(Prescribe strong painkillers A) = 1 Duration(Prescribe strong painkillers B) = 1 Duration(Prescribe weak painkillers) = 1 Duration(Prescribe anticoagulants) = 1 Duration(Prescribe anti-inflammatory drugs) = 1 Duration(Apply cast) = 15 Duration(Apply fixation) = 10 Duration(Apply bandage) = 5 Duration(Apply sling) = 3 Duration(Perform surgery) = 120 Duration(Let patient rest) = 180 Duration(Perform physiotherapy) = 60	Register patient requires RECEPTIONDESK and is executed by RECEPTIONIST Examine patient requires EXAMROOM and is executed by DOCTOR Take X-ray requires XRAYROOM and is executed by NURSE Apply cast requires EXAMROOM and is executed by NURSE or DOCTOR Apply fixation requires EXAMROOM and is executed by NURSE or DOCTOR Apply bandage requires EXAMROOM and is executed by NURSE or DOCTOR Apply sling requires EXAMROOM and is executed by NURSE or DOCTOR Perform surgery requires OPERATINGROOM and is executed by SURGEON Let patient rest requires PATIENTBED Perform physiotherapy requires PHYSIOTHERAPYROOM and is executed by PHYSIOTHERAPIST Prescribe weak painkillers is executed by DOCTOR Prescribe strong painkillers B is executed by DOCTOR Prescribe strong painkillers A is executed by DOCTOR Prescribe anti-inflammatory drugs is executed by DOCTOR Prescribe anticoagulants is executed by DOCTOR
Resource and role availabilities	
#RECEPTIONDESK = 3 #EXAMROOM = 15 #XRAYROOM = 1 #OPERATINGROOM = 4 #PATIENTBED = 60 #PHYSIOTHERAPYROOM = 2 #RECEPTIONIST = 3 #DOCTOR = 3 #NURSE = 10 #SURGEON = 2 #PHYSIOTHERAPIST = 1	

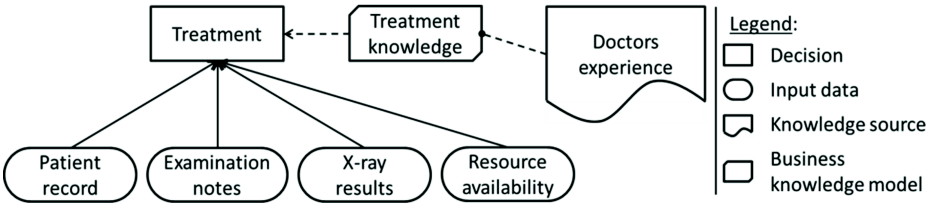


Fig. 2. Decision Requirements Graph for the arm fracture treatments

traceability as justification of decisions taken in actual cases [15]. Additionally, decision tables are also declarative [14], just like Declare, as the columns are rules with no particular order for conditions and actions to occur (i.e., stating the boundaries of the environment instead of a precise path through it). Lastly, decision tables can easily be annotated with statistical information to specify the likeliness of certain sets conditions, and thus of the chosen action [14].

Table 3. The decision table for the arm fracture treatments

Context: *Examine patient*									**Role:** *Doctor*										
Open or complex fracture	Y				N														
Extensive damage to arteries or nerves	-				Y			N											
Fractured bone	Wrist or Forearm		Finger, Hand, Upper arm or Shoulder	Collarbone	Wrist or Forearm		Finger, Hand, Upper arm or Shoulder	Collarbone	Finger or Hand	Collarbone	Forearm or Shoulder	Upper arm		Wrist					
Both arms broken	-		-	-	-		-	-	-	-	-	Y	N	-					
No improvement in 3 months	-		-	-	-		-	-	-	-	-	-	Y	N	-				
Age	-		-	-	-		-	-	-	-	-	-	-	-	0-16		>16		
Periosteum torn	-		-	-	-		-	-	-	-	-	-	-	-	Y	N	-		
OR available	Y	N	-	-	Y	N	-	-	-	-	-	-	-	-	-	-	-		
Apply fixation	-	-	-	-	-	-	-	-	X	-	-	-	-	X	-	-	-		
Perform surgery	X	-	X	X	X	-	X	X	-	-	-	X	X	-	X	-	-		
Apply cast	-	X	-	-	-	X	-	-	-	-	X	-	-	-	-	X	X		
Apply bandage	-	-	-	-	-	-	-	-	-	X	-	-	-	-	-	-	-		
response1	X	X	-	-	X	X	-	-	-	-	-	-	-	-	-	-	-		
response2	X	X	X	-	X	X	X	-	X	-	X	X	X	X	X	X	X		
response3	-	X	-	-	-	X	-	-	-	-	-	-	-	-	-	-	-		

Note that we added a context and role definition to each decision table. This specifies, respectively, the activities connected with the decision and the process actors that are responsible for taking it. The former provides us with a clear overview of what decisions are applicable during which activities, while the latter is a safeguard against unauthorized usage.

The decision table representation also has some drawbacks. When the decisions themselves are based on a very large amount of conditions and actions, the readability of a table gets lost. In such cases, the decision table will need to be split in multiple smaller tables to allow them to stay manageable. However, this comes at the price of cluttering the overall overview and understandability. Another problem arises when multiple actions are activated at the same time. Consider for example if the patient has a complex fracture of his forearm. The treatment for this diagnosis is surgery, followed by applying a cast. Decision tables do not allow a sequencing of these actions, but this is rather important in this situation. As a solution, we propose to introduce a new type of constraint in the Declare model: the *decision-dependent constraint*. It can embody all the templates from Table 1 and is visualized as a wavy line

in the model. The decision-dependent constraint represents a constraint that can only be activated as outcome of a decision table. If we retake the previous example, we can model this by adding a decision-dependent response constraint between 'Perform surgery' and 'Apply cast' (i.e., response1) and adding the additional outcome activating this constraint in Table 3. In a similar way, we add a decision-dependent response constraint between 'Examine patient' and 'Apply sling' (i.e., response2) and another decision table outcome activating this constraint to model the fact that in all treatments, except of collarbone fractures, the patient will receive a sling eventually. Lastly, another one of these constraints (i.e., response3) is used to ensure that surgery will still be performed later on when a temporary cast is applied because the OR is not available at that time.

Table 4. The decision table for the prescription of medication

Context: *Examine patient or Perform surgery*	**Role:** *Doctor or Surgeon*									
In pain (now or in foreseeable future)	Y									N
Age	0-16					>16				-
Previous step is surgery	Y		N			Y	N			-
On weak painkillers	Y	N	Y		N	-	Y		N	-
On anticoagulants or anti-inflammatory drugs	-	-	Y	N	-	-	Y	N	-	-
Prescribe strong painkillers A	-	-	-	X	-	-	-	X	-	-
Prescribe strong painkillers B	X	-	X	-	-	X	X	-	-	-
Prescribe weak painkillers	-	X	-	-	X	-	-	-	X	-
Prescribe anticoagulants	X	X	-	-	-	X	-	-	-	-
Prescribe anti-inflammatory drugs	X	X	-	-	-	X	-	-	-	-

Multiple outcomes in a decision table do not always lead to new sequencing constraints. Consider Table 4, which represents the decision to prescribe medication to patients. If the patient for example is in excessive pain, older than 16 years old and just had surgery, three actions are activated. The sequencing of these actions does not matter here, as they are just prescriptions (of course sequencing rules could apply when administering these drugs, but that is beyond the scope of the case).

Besides mandatory constraints, Declare allows for optional constraints to be modeled. We propose to extend this principle to the decision tables in Declare-R-DMN, as this allows them to represent 'wanted'-behavior (i.e., "should" and "ought to"). For example, patients that have had surgery, a cast applied and/or a fractured shoulder should do physiotherapy afterwards. But this is case dependent, so it should be possible to deviate from this structure. We modeled this in an optional decision table in Table 5. Following the representation in Declare, we visualized the 'optional'-property with a dotted line around the decision table. Table 6 does the same for the decision whether or not to take an X-ray.

Table 5. The optional decision table for the prescription of physiotherapy

Context: *Examine patient*	**Role:** *Doctor*			
Surgery was performed	Y	N		
Cast has been applied	-	Y	N	
Shoulder fracture	-	-	Y	N
Perform physiotherapy	X	X	X	-

Table 6. The decision table for an X-ray

Context: *Examine patient or Apply cast*	**Role:** *Doctor*			
Verifying fracture diagnosis	Y	N		
Fracture	-	Y		N
Verify if bone is correctly positioned under cast	-	Y	N	-
Take X-ray	X	X	-	-

5 Analysis of the Declare-R-DMN Language

Declare-R-DMN has the expressive power needed to create a more complete model, compared to Declare-R, of the process of the arm fracture case. This is achieved by adding support to Declare to model decision logic, without losing sight of the need for flexibility of KiPs. Making the decision logic explicit also facilitates the justification of taken decisions, better conformance checking and reuse [18] across different processes or even organizations. Value is created for the users and the organization as declarative process modeling languages are better suited for these types of processes compared to traditional process modeling languages. Additionally, more value is created by adding support for decision logic to the declarative modeling language as this is essential information that would otherwise be omitted.

Recently, other approaches have been proposed [17, 18] that do similar work. They add a way to model constraints that deal with the data aspect. However, these approaches focus primarily on the expressibility of the language and therefore much less on understandability and the modeling aspect. It is our opinion that these aspects are crucial for the adoption of the technique, and thus should be more of a priority. The use of decision tables (the DMN standard is committed to also offer support for other techniques in the near future) for this purpose is pretty straightforward, because they are a known and proven way of representing decisions and they are understandable for both technical as business people. By aligning the interpretation of the decision tables with Declare (i.e., decision-dependent constraints), adding context definitions and role responsibilities and extending the optionality-concept of Declare, Declare-R-DMN becomes a comprehensive and coherent modeling language: the temporal logic is modeled using Declare, the resource perspective using the Declare-R extension and the decision logic using DMN.

However, a general problem still persists. Compared to imperative modeling, the increased support for flexibility by declarative modeling comes at a price of understandability (of Declare in particular) and maintainability issues arise [3, 6, 19]. Recently, a couple of hybrid approaches have been proposed [20, 21] that offer some improvements for semi-structured processes, but not for unstructured processes.

6 Conclusion and Future Work

This paper presents an idea that is similar to what DMN attempts to do for BPMN, but also differs in a fundamental way. First, BPMN already somewhat supported decision logic with its fundamental concepts. Second, in the context of dynamic, knowledge-intensive and flexible processes, where Declare finds its niche, the decision logic is of much greater importance than in case of static and standardized processes. Decision logic is essential when modeling these processes as it offers valuable insight and encapsulates the knowledge of the domain experts executing the process.

The proposed language, Declare-R-DMN, combines Declare-R and DMN in a way that both original languages are supported as well as some new concepts that bridge them together (i.e., decision-dependent constraints, context definitions, role responsibilities and extending the optionality-concept). The usefulness of Declare-R-DMN was demonstrated by modeling a case example, representing a realistic example of a dynamic, knowledge-intensive and flexible healthcare process, as is exhibited by the large variety of possible execution paths (i.e., theoretically infinite). Where Declare-R did not offer enough tools to model the process of the arm fracture case, Declare-R-DMN thrived by incorporating the knowledge that is essential to the case.

The scope of this paper was limited to a general elaboration of the idea. In the next phase we will formalize the semantics and metamodel of this new language. For this purpose, we need to analyze and propose solutions for all of its possible ambiguities, overlaps and shortcomings to obtain a clear and coherent language. Inspiration can come from the proposals for a data-aware Declare [17, 18]. The language will then be further evaluated by using it to model similar real-life cases from different areas.

References

1. Goedertier, S., Vanthienen, J., Caron, F.: Declarative business process modelling: principles and modelling languages. Enterp. Inf. Syst., 1–25 (2013)
2. Lu, R., Sadiq, W.: A Survey of Comparative Business Process Modeling Approaches. In: Abramowicz, W. (ed.) BIS 2007. LNCS, vol. 4439, pp. 82–94. Springer, Heidelberg (2007)
3. Haisjackl, C., Barba, I., Zugal, S., Soffer, P., Hadar, I., Reichert, M., Pinggera, J., Weber, B.: Understanding Declare models: strategies, pitfalls, empirical results. Softw. Syst. Model. (2014)
4. Krogstie, J.: Perspectives to Process Modeling – A Historical Overview. In: Bider, I., Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Soffer, P., Wrycza, S. (eds.) EMMSAD 2012 and BPMDS 2012. LNBIP, vol. 113, pp. 315–330. Springer, Heidelberg (2012)

5. Van der Aalst, W.M.P., Pesic, M., Schonenberg, H.: Declarative workflows: Balancing between flexibility and support. Comput. Sci. - Res. Dev. **23**, 99–113 (2009)
6. Pesic, M.: Constraint-based workflow management systems: shifting control to users (2008)
7. Barba, I., Del Valle, C.: Filtering rules for ConDec templates - Pseudocode and complexity. http://www.lsi.us.es/quivir/irene/FilteringRulesforConDecTemplates.pdf (accessed on October 9, 2014)
8. Payton, F.C., Paré, G., LeRouge, C., Reddy, M.: Health care IT: Process, people, patients and interdisciplinary considerations. J. Assoc. Inf. Syst. 12 (2011)
9. Boxwala, A.A, Peleg, M., Tu, S., Ogunyemi, O., Zeng, Q.T., Wang, D., Patel, V.L., Greenes, R.A, Shortliffe, E.H.: GLIF3: a representation format for sharable computer-interpretable clinical practice guidelines. J. Biomed. Inform. 37, 147–61 (2004)
10. Seyfang, A., Kosara, R., Miksch, S.: Asbru's Reference Manual v7.3 (2002)
11. Fox, J., Johns, N., Rahmanzadeh, A.: Disseminating medical knowledge: the PROforma approach. Artif. Intell. Med. **14**, 157–181 (1998)
12. Mulyar, N., Pesic, M., van der Aalst, W.M., Peleg, M.: Declarative and Procedural Approaches for Modelling Clinical Guidelines: Addressing Flexibility Issues. In: ter Hofstede, A.H., Benatallah, B., Paik, H.-Y. (eds.) BPM Workshops 2007. LNCS, vol. 4928, pp. 335–346. Springer, Heidelberg (2008)
13. OMG: Decision Model and Notation (DMN). www.omg.org/spec/DMN/Current/ (accessed on November 10, 2014)
14. Software Testing Genius: Decision Table Based Testing-Black Box Software Testing Technique. http://www.softwaretestinggenius.com/decision-table-based-testing-black-box-software-testing-technique
15. Decision Table Task Group: A Modern Appraisal Of Decision Tables (1982)
16. Zugal, S., Soffer, P., Haisjackl, C., Pinggera, J., Reichert, M., Weber, B.: Investigating expressiveness and understandability of hierarchy in declarative business process models. Softw. Syst. Model. (2013)
17. Montali, M., Chesani, F., Mello, P., Maggi, F.M.: Towards data-aware constraints in declare. In: Proceedings of the 28th Annual ACM Symposium on Applied Computing, SAC, pp. 1391–1396. ACM Press, New York (2013)
18. Borrego, D., Barba, I.: Conformance checking and diagnosis for declarative business process models in data-aware scenarios. Expert Syst. Appl. **41**, 5340–5352 (2014)
19. Zugal, S., Pinggera, J., Weber, B.: Toward enhanced life-cycle support for declarative processes. J. Softw. Evol. Process., 285–302 (2012)
20. De Smedt, J., De Weerdt, J., Vanthienen, J.: Multi-Paradigm Process Mining: Retrieving Better Models by Combining Rules and Sequences. SSRN Electron. J. (2014)
21. Maggi, F.M., Slaats, T., Reijers, H.A.: The Automated Discovery of Hybrid Processes. In: Sadiq, S., Soffer, P., Völzer, H. (eds.) BPM 2014. LNCS, vol. 8659, pp. 392–399. Springer, Heidelberg (2014)
22. Reichert, M., Weber, B.: Enabling flexibility in process-aware information systems. Springer (2012)