Bus Inf Syst Eng 58(5):355–366 (2016)
DOI 10.1007/s12599-016-0443-3

RESEARCH PAPER

Supporting the Refinement of Clinical Process Models to Computer-Interpretable Guideline Models

Begoña Martínez-Salvador · Mar Marcos

Received: 4 June 2014 / Accepted: 29 January 2016 / Published online: 9 August 2016

Abstract Clinical guidelines contain recommendations on the appropriate management of patients with specific clinical conditions. A prerequisite for using clinical guidelines in information systems is to encode them in a Computer-Interpretable Guideline (CIG) language. However, this is a difficult and demanding task, usually done by IT staff. The goal of the paper is to facilitate the encoding of clinical guidelines in CIG languages, while increasing the involvement of clinicians. To achieve this, it is proposed to support the refinement of guideline processes from a preliminary specification in a business process language to a detailed implementation in one of the available CIG languages. The approach relies on the use of the Business Process Model and Notation (BPMN) for the specification level, a CIG language for the implementation level, and on algorithms to semi-automatically transform guideline models in BPMN into the CIG language of choice. As a first step towards the implementation of the approach, in this work algorithms are implemented to transform a BPMN specification of clinical processes into the PROforma CIG language, and are successfully applied to several clinical guidelines.

Keywords Clinical guideline representation · BPMN · PROforma · Transformation between process languages

Accepted after two revisions by Prof. Dr. Jarke.

B. Martínez-Salvador (✉) · M. Marcos
Department of Computer Engineering and Science, Universitat Jaume I, Av. de Vicent Sos Baynat s/n, 12071 Castellón, Spain
e-mail: begona.martinez@uji.es

M. Marcos
e-mail: marcos@uji.es

1 Introduction

Clinical guidelines are defined as “systematically developed statements to assist practitioner and patient decisions about appropriate health care for specific circumstances” (Field and Lohr 1990). Guidelines contain evidence-based recommendations for the best management of patients with a particular clinical condition. Clinical guidelines improve the process and the outcome of healthcare. For example, they support evidence-based medicine, reduce variability in the application of the procedures and also decrease the possibility of errors (Boxwala et al. 2001). Clinical guidelines are usually text documents, sometimes augmented with more structured information like flowcharts to specify some recommendation steps.

The best way to implement the clinical guideline at the point of decision-making, when the patient-clinician encounter occurs, is by implementing an alert-based system or a more complex decision-support system. A prerequisite for implementing such systems is to transform the textual guideline into a computer-interpretable format, that is, into a Computer-Interpretable Guideline (CIG). For this purpose, in the Medical Informatics area, several languages for modeling CIGs have been developed. The most important languages for CIGs are (Peleg et al. 2003; de Clercq et al. 2004): Arden Syntax, Asbru, EON, GLIF, GLIDE, Prodigy and PROforma. These languages are tailored to the singularities of the medical domain. They share many features, although each one has its own characteristic elements. Inspite of some attempts in this direction, no CIG language has become a standard.

In practice, it turns out that encoding the recommendations of a clinical guideline (mainly, its clinical processes) in a CIG language is a demanding task that requires both clinical and IT skills. On the one hand, clinical knowledge is required for a proper understanding of most of the recommendations in clinical guidelines. On the other hand, IT skills are required to analyze the clinical processes they contain and to describe them in terms of a CIG language. This is because CIG languages are not always accessible for clinicians. Our goal is to facilitate the encoding of clinical guidelines in CIG languages, while increasing the involvement of clinicians in the process. To achieve this, we propose to support the refinement of clinical processes in guidelines from a preliminary specification in a business process language to a detailed implementation in one of the available CIG languages.

Concretely, our approach relies on the use of the Business Process Model and Notation (BPMN) for the specification level, a CIG language for the implementation level, and on algorithms to semi-automatically transform guideline models in BPMN into the CIG language of choice. Because of the latter, we designate our approach as *transformation-based refinement*. Compared to the direct encoding in the CIG language, our approach supposes an initial BPMN modeling step plus a semi-automatic transformation step. An important advantage of the initial modeling lies in its potential to increase the involvement of clinicians. As a matter of fact, we envisage that this step will be mainly performed by clinicians, with the assistance of IT staff. Another advantage is that the effort to model a clinical process in BPMN can be leveraged by the implementation of models in several CIG languages, provided that appropriate transformation methods are developed.

Clinical processes can be represented using a standard process modeling language, such as BPMN. Because the last BPMN specification (OMG 2011) provides some execution semantics in terms of BPEL, in general BPEL is mistaken for an executable expression of BPMN. However, the full equivalence of BPMN cannot be expressed in BPEL (Dugan and Palmer 2012). For this reason we do not regard BPMN as implementation language, but rather as an initial specification that can be used as a basis for a later implementation. BPMN has rapidly earned wide acceptance, becoming a de facto standard for graphical process modeling (Recker 2010). To date most users have employed BPMN to describe operations in a simple and graphical way. The situation is similar in the medical domain. Some works have used BPMN for the collaborative modeling of clinical pathways (Kirchner et al. 2014; Scheuerlein et al. 2012), resulting in higher quality models which are better understood and accepted by domain experts. All this supports the use of BPMN as an instrument for the preliminary specification of processes in clinical guidelines.

As implementation language, any of the aforementioned CIG languages may be chosen. In this work, we target PROforma, primarily due to our previous modeling experience with this language. PROforma is one of the most important languages for CIGs, and is actively supported by OpenClinical.org, a community of healthcare professionals and medical informatics researchers. Moreover, there are several software tools available to work with PROforma guidelines, such as a graphical editor, a tester, and a web-enactment suite.

As a first step towards the implementation of our transformation-based refinement approach, in this article we describe the algorithms that we have implemented for the transformation of guideline models specified in BPMN into the PROforma language. A preliminary description of the algorithms was introduced in Martínez-Salvador et al. (2014). The transformation algorithms have been tested with different guidelines. As an illustration, some results obtained with a guideline for the diagnosis and treatment of prostate cancer (Mohler et al. 2012) are presented.

The rest of the article is structured as follows. Section 2 presents an overview of BPMN, PROforma, and the methods. Section 3 is devoted to the implementation of the transformation algorithms. In Sect. 4, some experimental results with a prostate cancer guideline are presented. Finally, Sect. 5 concludes and outlines some future work.

2 Materials and Methods

2.1 BPMN

The Object Management Group (OMG) has developed the BPMN notation which provides a standard graphical notation for specifying business processes. The latest published specification is BPMN 2.0 (OMG 2011).

In recent literature we can find several works using BPMN in the medical domain. Some works report experiences in modeling processes for patients with a particular condition (Rojo et al. 2008; Rolón et al. 2008; Parra et al. 2012). Others use BPMN for modeling clinical pathways (Svagård and Farshchian 2009; Scheuerlein et al. 2012; Hashemian and Abidi 2012; Kirchner et al. 2014). Most of the works agree on emphasizing that BPMN is easy to use and understand by all stakeholders. In one of the works BPMN is used to model the anatomic pathology processes in a Spanish hospital (Rojo et al. 2008). The modeling team comprised external IT experts and hospital staff, including health experts and people responsible for administrative and quality issues. The authors conclude that the resulting model is understandable for involved health professionals, and that it improves communication. There are also works that describe the experiences in collaborative modeling of clinical pathways by health professionals assisted by IT staff. They report that familiarization with BPMN is relatively quick and intuitive

(Scheuerlein et al. 2012), and that the fact that health experts have a better understanding of clinical pathways facilitates modifications and updates of the model (Kirchner et al. 2014).

A BPMN process describes a flow of activities in an organization with the objective of carrying out a task. It is depicted as a Business Process Diagram (BPD). BPMN is a complex language with many graphical elements. However, a study conducted by zur Muehlen and Recker (2008) showed that the average BPMN model uses less than 20 % of the available elements. In the rest of the section, we restrict our discussion to a subset of BPMN elements: events, gateways, tasks, sub-processes and sequence flows. These elements roughly include the BPMN common core and extended core defined by Recker (2010), except for the pool and lane elements. With this subset, it is possible to specify real-world clinical guidelines as the one used in this paper (see Sect. 4).

BPMN flow objects are the main elements for defining the behavior of a business process. There are three types of flow objects: *activities*, *events*, and *gateways*. BPMN also has connecting objects which are used to connect flow objects to each other or to data objects. The main type of connecting object are *sequence flows*.

An event is something that “happens” during the course of a process. The *start event* indicates where a particular process will start. Similarly, the *end event* indicates where a particular process will finish.

Gateways control branching and merging of flows in a process. The *gatewayDirection* might be set to *converging* or *diverging*. If it is set to converging, then the gateway must have multiple incoming flows and only one outgoing flow. Reciprocally, if it is set to diverging, the gateway cannot have multiple incoming flows but must have multiple outgoing flows. There are different types of gateways to control the flow behavior. A diverging *exclusive* gateway (split XOR-gateway) is used to create alternative paths within a process flow. A diverging *inclusive* gateway (split OR-gateway) is used to create alternative paths where more than one of them can be followed. A diverging *parallel* gateway (split AND-gateway) is used to create parallel flows. We will refer to converging gateways as join gateways, e.g., a join XOR-gateway.

Activities are points in the process where work is performed. There are two types of activities: *tasks* and *sub-processes*. A task is an atomic activity. It represents an action that is not further refined. BPMN specifies three types of markers for tasks. In the domain of clinical guidelines, we have used the *loop* marker which indicates that the task may be repeated. The number of iterations depends on a condition that is evaluated for each iteration.

A sub-process is an activity whose internal details have been modeled in another BPD. The nested elements are represented collectively as a single activity in the diagram. Sub-processes are used to hide the complexity of a diagram or to define a special way of execution for the activities within it. *Ad-hoc sub-processes*, a special type of sub-processes, have an ordering attribute whose value can be set to sequential or to parallel. If it is set to sequential, the inner processes will be executed in every possible sequential arrangement. If it is set to parallel, it is possible to have several processes or activities enacted at the same time. Sub-processes, as tasks, may also have a loop marker.

Sequence flows indicate the order in which activities will be performed. Each sequence flow must have exactly one source and one target flow object. Sequence flows coming of from split gateways optionally define a condition expression to be evaluated before deciding whether or not to follow that flow. In the case of split XOR and OR-gateways it is possible to define a *default* sequence flow, which will indicate the path to be chosen in case all the condition expressions evaluate to false.

Figure 1 shows the specification of the main clinical process for the diagnosis and treatment of prostate cancer. The figure contains events, XOR-gateways, tasks and collapsed sub-processes which contain their own BPD.

2.2 PROforma

PROforma is an executable CIG language, tailored to capture medical knowledge and successfully used for deploying clinical decision support systems (Sutton and Fox 2003). The Tallis implementation (COSSAC 2013b) provides a Composer for authoring CIG-based decision support systems that may be enacted using the Tallis Engine. There are several examples of decision support systems implemented in PROforma such as Bury et al. (2005), Coulson et al. (2001), Emery et al. (2000), and COSSAC (2013a).

The building blocks of PROforma are the *tasks*. Tasks represent actions or activities to be performed by an external agent (e.g., clinician, patient) or by the Tallis engine itself. There are four types of *tasks*: *Enquiry*, *Decision*, *Action* and *Plan*. Enquiries request data from the environment, to be entered by a human user or read from a database. Decisions are activities where a choice has to be made among different options. Actions represent those activities that have to be performed in the external environment (e.g., perform blood glucose level test). Finally, plans group together any type of tasks.

Control flow is represented in PROforma by means of *scheduling constraints*. Scheduling constraints are logical expressions that determine in which order the tasks should be enacted. Graphically, scheduling constraints are represented by directed arcs. The direction indicates that one task (the one at the head of the arc) cannot start until

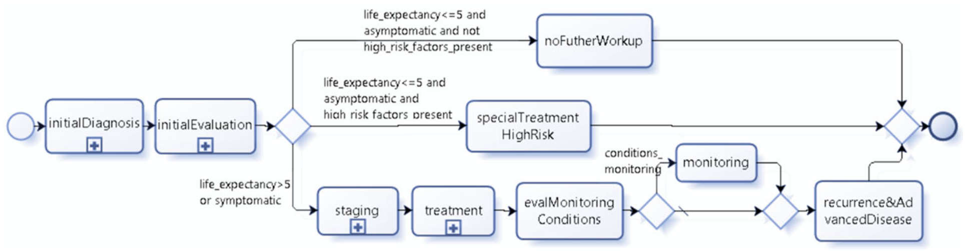

Fig. 1 Main clinical process in BPMN for the diagnosis and treatment of prostate cancer. Events are depicted as *circles*. *Rounded boxes* are tasks, or sub-processes if they have the mark +. Gateways are depicted as *diamonds* and sequence flows as *arrows*

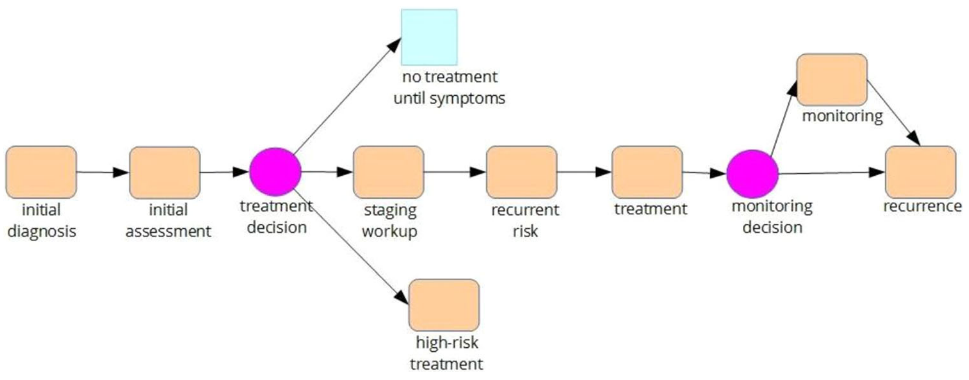

Fig. 2 PROforma graphical notation of the main clinical process for the diagnosis and treatment of prostate cancer. Plans are depicted as *rounded boxes*, actions as *squares* and decisions as *circles*. *Arcs* between tasks represent scheduling constraints

another task (the one at the tail) has completed. Naturally, a task can have more than one scheduling constraint.

Besides scheduling constraints, PROforma tasks may have a *precondition*, which is a truth-valued expression that must be satisfied when the task is started, and a *trigger precondition*, a truth-valued expression which will initiate a task if it is satisfied. Tasks can be cyclical. The number of iterations can be determined by an integer or by a truth-valued expression (*cycleUntil*).

Decisions are tasks in which a choice is made among several different options, known as *candidates*. Candidates have also properties of their own: zero or more *arguments*, and a *recommendation rule*. An argument is a truth-valued expression representing the arguments for or against a particular candidate. When a decision is enacted, the expression and support type (for or against) of all arguments are used to calculate the net support for the candidate. Basically, a for-argument adds one to the net support of the candidate, and an against-argument subtracts one. A recommendation rule is an expression that states the conditions under which it would be appropriate to commit to the candidate, based on the calculated net support.

Plans have additional control flow properties. A *termination condition* is a truth-valued expression which represents the sufficient condition to successfully terminate the plan. An *abort condition* is a truth-valued expression that aborts the plan.

Lastly, enquiries have *sources*, which are data items whose value has to be supplied. An enquiry may define several sources and each source is based on a *data definition*.

Process descriptions are modeled in PROforma using the set of tasks and logical constructors. From the initial root plan, tasks are organized hierarchically into plans. Figure 2 shows the PROforma graphical representation of the main clinical process of the same guideline shown in Fig. 1.

2.3 Approach

The transformation of a clinical guideline specification in BPMN to a CIG can be approached as a transformation between two different modeling languages.

There are several works which address the transformation from BPMN to BPEL by means of algorithms (Mendling et al. 2008; Ouyang et al. 2009). These papers exploit the graph-oriented paradigm of BPMN in order to implement generic strategies for transforming graph-oriented process languages into block-oriented ones (such as BPEL). One of the transformation strategies is what those authors call a *structure-identification strategy*.

Starting from certain structures of interest in the target language, this strategy consists in identifying in the source language structures that are equivalent to those structures of interest. Then, each identified structure is mapped to the target language and replaced by a single node according to the reduction rules applied in the definition of structured process graphs (Mendling et al. 2008). A very similar approach is used for transforming XPDL to a Hierarchical Task Network (HTN) (González-Ferrer et al. 2013). The main advantage of the structure-identification strategy is that it produces more readable and understandable target code. This strategy is only applicable to structured and acyclic input models.

In order to apply the structure-identification strategy, the input process graph is segmented into proper structures. Few papers address the necessary graph segmentation. We have studied two approaches: the token analysis algorithm (Götz et al. 2009) and the branch-water algorithm (Bae et al. 2004). Both algorithms have two main phases. The first phase consists in traversing the graph while labelling its nodes. In the second phase every structure of interest is identified and replaced by a single node in the graph. A component is a connected sub-graph, with at least two nodes, with a single entry point and a single exit point, and without start and end events. Gotz et al. and Bae et al. decompose the source graph into serial components, i.e. sequences, and parallel components. The entry point of a parallel component is a split gateway, and the exit point is the corresponding join gateway. Parallel components comprise OR-parallel components, XOR-parallel components, and AND-parallel components, according to the different types of gateways.

Although the PROforma language is not a BPM language, it has features of graph-oriented and block-oriented paradigms. In this work, we exploit the graph-oriented features of the input model and detect suitable structures that are then translated to PROforma elements. Moreover, our approach is tailored to the characteristics of clinical processes.

The so-called workflow patterns (Van der Aalst et al. 2003) are somehow related to the above-mentioned structures of interest, as it is possible to recognize a workflow pattern in some of them. However, these structures are determined exclusively on the basis of the target language elements, which may differ considerably from the workflow patterns (see Sect. 3.2). Additionally, our aim is to exploit the graph-oriented features of BPMN, and to produce readable and understandable target code.

3 Transformation to PROForma

3.1 The Input Model

In the context of clinical guidelines, our input BPMN models have several important features that have been taken into account in the implementation of the transformation algorithms. First and very importantly, the BPMN input models we consider are structured process models. A structured model is one in which every split gateway has a matching join gateway of the same type, and in which all split-join pairs are properly nested (Kiepuszewski et al. 2000). Since clinical guidelines are formulated in natural language, non-structuredness is neither an essential nor useful feature for clinical process models. This is an asset since the structure-identification strategy is only applicable to structured graphs. Moreover, structuredness is a desirable property of BPDs according to Mendling et al. (2010).

Moreover, our input models use the BPMN elements sub-process and ad-hoc sub-process, for the following reasons. Communication and clarity are among the most important purposes of BPMN, also in the case of guideline processes. However, BPMN models with a high number of elements are difficult to understand and more error-prone. Therefore, by convention, we split up complex BPDs with a large number of elements into smaller and simpler BPDs with sub-processes hiding the internal details of certain activities (see e.g., Fig. 1). This is in line with one of the seven process modeling guidelines by Mendling et al. (2010).

Additionally, clinical guidelines may contain recommendations that can be modeled as iterative processes. We have considered the usage of loops in tasks or sub-processes for modeling this type of processes. However, we do not consider arbitrary cycles in our models.

Finally, we regard a particular type of XOR-parallel components. When gateways are used in BPMN models, normally at least one activity takes place in all paths between the split and join gateway. According to our experience, clinical guidelines frequently contain recommendations to be applied only to a subgroup of patients. For example, the guideline for the diagnosis of prostate cancer recommends a bone scan, a tomography or a MRI for certain patients, while no additional imaging is recommended for the rest of patients. This kind of recommendations are usually modeled with a sequence flow that directly connects the split with the join XOR-gateway, as Fig. 3 shows.

3.2 The Mapping to PROforma

In order to apply the structure-identification strategy, we have studied the building blocks of the target language, in this case PROforma. These comprise plans, decisions, scheduling constraints, and actions. Plans group processes but are also used to model parallel flows. Then, it is necessary to identify AND-parallel components in the input BPMN graph in order to transform them into PROforma plans. The identification of AND-parallel components is done based on the split and join AND-gateways delimiting them. Notice that these AND-gateways need not be mapped since parallelism is represented in PROforma by means of a plan, which means that PROforma is more compact in this case.

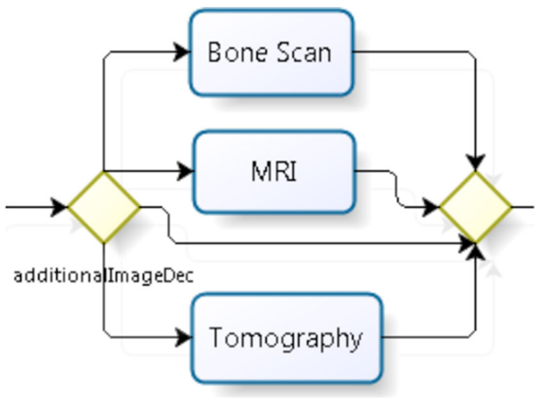

Fig. 3 XOR-parallel component with an arc that directly connects the split with the join gateway, modeling the option "no imaging for some patients"

PROforma decisions model the control structures if-then, pick one, and pick one or more. In the input model, these patterns correspond either to XOR or OR-parallel components. Therefore, XOR and OR-parallel components must be identified to be transformed to PROforma decisions. To facilitate the transformation, and given that there can be any type of elements including other components between the split and join gateway of an XOR or OR-parallel component, we have transformed every such component into a PROforma plan with a decision task inside. A PROforma decision needs to define its candidates and the arguments for/against these candidates (see Sect. 2.2). In general, the successor nodes of the split XOR-gateway will result in the candidates, although the transformation also works with XOR-parallel components like the one in Fig. 3. The condition expressions of the outgoing sequence flows of the split gateway will define the arguments for each candidate. Finally, a text analysis of these condition expressions will provide the sources (data) of the PROforma decision.

In PROforma, scheduling constraints are the way of specifying the order in which tasks are enacted. In the case of sequences, there is a scheduling constraint between each pair of consecutive tasks. Thus, sequences are identified in the input process graph and are translated to appropriate scheduling constraints. A scheduling constraint connects two consecutive tasks, and is graphically represented by a directed arrow connecting those tasks. We could say that plans and decisions represent the block-oriented features of PROforma, while scheduling constraints are the graph-oriented ones.

Finally, every BPMN task will be translated into a PROforma action, and every sub-process will be mapped to a PROforma plan with the aim of maintaining the same process grouping. There are attributes, such as the loop condition, that will be mapped to a PROforma attribute.

Table 1 Mappings between the BPMN elements and PROforma elements

BPMN	PROforma
XOR-parallel component	Plan with decision
OR-parallel component	Plan with decision
AND-parallel component	Plan
Sequential component	Scheduling constraints
ConditionExpression in SequenceFlow	Argument in decision
Successor node of split XOR/OR-gateways	Candidate in decision
Variable in ConditionExpression	Source in decision
Task	Action
Parallel ad-hoc sub-process	Plan
Sequential ad-hoc sub-process	Plan with decision + plans
Sub-process (non ad-hoc)	Plan
Loop expression	cycleUntil
AND gateway	–
Any join gateway	–
Start event	–
End event	–

The mappings are listed in Table 1. The correspondence between BPMN and PROforma elements is not always one-to-one. This is because PROforma representation is more compact in some cases, as illustrated by the case of AND-parallel components. This implies that some BPMN elements will not be taken into account in the mapping to PROforma. Note that the fact that the representation in PROforma is more compact in some cases does not necessarily make the language less precise. In fact, the degree of detail in the PROforma representation is at least the same as the degree of detail in the BPMN one. At the other extreme, PROforma representation of decisions (XOR and OR-parallel components) requires a considerable level of detail.

3.3 Implementation

The implementation of the approach has three steps which are: (1) storing the BPMN model on a graph data structure; (2) segmenting the graph, which includes graph labelling, component identification and graph reduction; and finally, (3) generating the target code. The algorithms have been implemented in Java using the open-source Java JDOM API[1] for manipulating XML data.

The first step is to build a directed graph from the BPMN specification of the clinical procedures. Thus, every BPMN flow object is represented by a node in the graph

[1] http://www.jdom.org/docs/apidocs/org/jdom2/input/SAXBuilder.html. Accessed 13 June 2013.

and every sequence flow becomes an arc in the graph. Since we deal with sub-processes, we have a graph of graphs. In other words, we work with a recursive data structure where every sub-process is represented as a single node in the graph, while it contains its own graph and possibly sub-graphs.

Moreover, this data structure has been enriched with additional information which is read from the BPMN file. This information includes the type of activity, the type of gateway, the timing of activities and the type of condition and the condition itself, if any. Conditions might be associated to activities or to sequence flows.

In this step, there is a pre-processing of the graphs included in ad-hoc sub-processes. Since the inner processes of a parallel ad-hoc sub-process can be enacted simultaneously, we have modeled them using an AND-gateway with an arc for each inner process. On the other hand, to mimic the behavior of a sequential ad-hoc sub-process, each possible sequential arrangement of the inner processes has been modeled as an alternative after a split XOR-gateway. Note that these sequential arrangements are not part of the initial BPMN model. In this sense, we can say that these components have been artificially created. Considering the preprocessing, the mapping of sequential ad-hoc sub-processes to PROforma includes a plan with a decision and several subplans, as Table 1 shows. This mapping adds complexity to the resulting model, but cannot be avoided because there is no equivalent in PROforma for the BPMN sequential ad-hoc sub-process.

3.3.1 Segmenting the Graph

Graph segmentation into components is a key step in the transformation algorithm. We have adapted the branch-water algorithm to the features of our input graphs. Therefore, in order to deal with sub-processes, we have implemented a recursive solution.

The algorithm first labels all the vertices of the graph. It assigns an initial value (1.0) to the first node of the graph and propagates it through the graph. If a node splits the flow into several branches, the value is divided by the number of branches and propagated to the subsequent nodes. Conversely, the value of a node with several incoming arcs is calculated as the sum of the labels of the precedent nodes.

The labelling method has been adapted to deal with the type of sub-graphs shown in Fig. 3. We define the concept of *valid successor node* as follows: given a node representing a split gateway, a subsequent node is said to be a valid successor if it is not the corresponding join gateway. Reciprocally, with regard to a join gateway, we say that a precedent node is a *valid predecessor node*, if it is not the corresponding split gateway. Thus, the labels propagated through the arcs are calculated considering only the valid successor nodes. Likewise, the label of a join gateway is calculated from the values of the valid predecessor nodes.

Once all the nodes have been labelled, the algorithm proceeds identifying components, that is, sequences and parallel components. Each time a component is identified, its type and content are registered, and it is replaced by a single component node. In the end, the graph is reduced to a trivial graph that gives rise to a tree structure of components. In our work, we postpone the mapping to PROforma until the tree of components is obtained. Therefore, all components are identified first.

The branch-water algorithm always uses the minimum value of the set of labels to find the innermost component. Each time, starting at the beginning node, the graph is traversed until the innermost component is found. Thus, although the size of the graph decreases at each iteration, the graph is traversed several times.

Our implementation first traverses the graph once, identifies all sequences and replaces each one by a single node. After that, the algorithm iterates looking for the innermost parallel component which is replaced by a single node. And then, it looks for a possible new sequence considering the node created as a replacement of the parallel component.

Algorithm 1 shows the pseudo-code for the main algorithm. This algorithm calls Algorithms 2 and 3. Algorithm 2 detects and replaces all sequences with at least two nodes. Notice that only *maximal sequences* are of interest. A maximal sequence is defined as a series of consecutive nodes with the same label, excluding gateways and start and end events, such that it is not possible to add a new node to the existing sequence without loosing its features. Since several arcs merge in a join gateway, nodes are marked to avoid repeating the search of already identified sequences following a join gateway. Finally, Algorithm 3 shows the pseudo-code that seeks for a parallel component.

```
Algorithm 1: ComponentIdentificationAlg
  Input: Graph, Weights, GatewayNodes
  Output: Trivial graph, SequenceComponents, ParallelComponents
  allSequencesSearch(start node, Graph, SequenceComponents) ;
  while (non trivial graph) do
      parallelComponent ← parallelComponentSearch(Graph, Weights, GatewayNodes,
        ParallelComponents) ;
      if (ParallelComponent done) then
          add(ParallelComponents, parallelComponet) ;
          parallelComponentNode ← replace(Graph, parallelComponent) ;
          serialComponentSearch(Graph, parallelComponentNode,
            SequenceComponents) ;
      end
  end
  return Graph, SequenceComponents, ParallelComponents ;
```

```
Algorithm 2: allSequencesSearch
  Input: Graph, start node, SequenceComponents
  Output: Graph modified, SequenceComponents
  sequence = $\emptyset$ ;
  v = start node ;
  while (v is not a gatewayNode and v is not endEventNode) do
      if (v is a subprocessNode) then
          ComponentIdentificationAlg(v.graphmodel) ;
      end
      add(sequence,v);
      mark(v,visited) ;
      v = succ(v) ;
  end
  if ($|sequence| > 1$) then
      add(SequenceComponents, sequence) ;
      sequenceComponentsNode ← replace(Graph,sequence);
  end
  sequence = $\emptyset$ ;
  if (v is a diverging gatewayNode) then
      mark (v,visited) ;
      for (every node w in succ(v)) do
          allSequencesSearch(Graph, w, SequenceComponents) ;
      end
  else
      (v is a converging gatewayNode) ;
      if (v is not visited) then
          mark (v,visited);
          allSequencesSearch(Graph, succ(v), SequenceComponents);
      end
  end
  return Graph, SequenceComponents;
```

```
Algorithm 3: parallelComponentSearch
  Input: Graph, Weights, GatewayNodes
  Output: Graph modified, ParallelComponents, parallelComponentNode
  parallelComponent = $\emptyset$ ;
  min_w = minimum(Weights) ;
  v = node with min_w(GatewayNodes) ;
  if (found v) then
      add (parallelComponent, succ(v)) ;
      add (parallelComponent, join gateway) ;
      add (ParallelComponents, parallelComponent) ;
      parallelComponentNode ← replace (Graph, parallelComponent) ;
  end
  return Graph, parallelComponentNode, ParallelComponents
```

3.3.2 Generating the Target Code

Algorithm 1 results in a trivial graph, with just one node plus the start and end events. From this point, the replacement of each node by its content gives rise to a tree structure. The generation of the PROforma code is done following a top-down traversal of this tree. We start with the single node of the trivial graph and translate it to PROforma according to the mappings listed in Table 1. Then we replace the node by its content and the same process is repeated for each one of the new nodes.

The transformation to PROforma is done in two traversals of the tree. In the first traversal, the mapping of each node to PROforma is stored in the node itself, with the exception of the scheduling constraints. In the second traversal the mapping is written into a file, and the scheduling constraints are defined in this second traversal of the tree.

4 Experiments with a Prostate Cancer Guideline

We have conducted some experiments with different clinical guidelines. One of them is the NCC Prostate Cancer Guideline (Mohler et al. 2012), which is a 69-page text document with evidence-based recommendations for the diagnosis and treatment of prostate cancer. Prostate cancer is one of the most important causes of mortality and the most common cancer among males in developed countries (Siegel et al. 2013). We had previously modeled this guideline both in BPMN (Fig. 1) and in PROforma (Fig. 2), to familiarize our clinical collaborators with the notations for describing clinical processes and to gather their impressions.

In this section, we discuss the results of applying the transformation algorithms to the BPMN specification of the NCC prostate cancer guideline. On the one hand, we have manually checked that all the input model components were appropriately translated, according to the transformations defined for the PROforma structures of interest. On the other hand, we have executed a series of tests to ensure that the obtained PROforma model – which we refer to as *transformed model* – produced the intended results. That is, we have checked that for a series of test cases, the execution of the transformed model produces the same results that would be obtained by applying the text version of the guideline. This is the usual procedure we use for testing our models. A formal verification of the models is out of the scope of this work, as advanced techniques specific to some CIG languages are already available, including Asbru (Marcos et al. 2003) and PROforma (Grando et al. 2012). Finally, we have manually compared the transformed model with the version we had previously modeled in PROforma – which we refer to as *direct model* – to determine the equivalence of corresponding structures.

Table 2 Comparing the input BPMN model and the transformed PROforma models

Source BPMN		Transformed PROforma	
Size	325	Size	376
Depth	4	Depth	8
XOR-split gateways	49	Plans with a decision	57
OR-split gateways	6		
Seq. ad-hoc sub-processes	2	Plans	74
Paral. ad-hoc sub-processes	9		
Sub-processes	38		
Tasks	166	Actions	188

Table 2 shows, in the left-hand column, what BPMN elements the input model includes and in what quantities. The table also shows, in the right-hand column, the counterpart elements in the transformed PROforma model and their quantities. The BPMN model consists of 325 nodes, of which 49 are sub-processes, 55 are split gateways, 55 are the corresponding join gateways, and 166 are tasks. The transformed PROforma model consists of 376 elements. It has 57 plans that include a decision, which correspond to the same number of XOR and OR-parallel components. Of the total of 74 plans, 49 correspond to the same number of BPMN sub-processes. The rest correspond to the XOR gateways and the different alternatives introduced in the preprocessing of sequential ad-hoc sub-processes (see Sect. 3.3). For the same reason, the number of PROforma actions is greater than the number of BPMN tasks.

Apart from the number of plans and actions, the most noticeable difference is in the depth of the models. Both BPMN and PROforma models define a hierarchical structure, based on sub-processes and plans, respectively. The depth of the transformed model is 8, i.e., twice the depth of the BPMN model. This is due to the transformation of XOR-parallel components. Another interesting observation regards BPMN condition expressions, which can be just represented as plain text. This is very convenient if conditions are to be used for annotation purposes in the specification phase. In our BPMN guideline the modeler has carefully written these expressions, which allows the program to properly parse them and extract the data items required for the decision tasks. However, it cannot be presupposed that all data items can always be extracted in this way, and therefore a manual revision of data sources will be required.

Table 3 shows the number of elements in the direct and in the transformed PROforma models. The direct model has a total of 246 elements divided into 61 plans, 21 decisions, 105 actions, and 59 enquiries. Here again, the

Table 3 Comparing the direct and the transformed PROforma model

	Direct PROforma	Transformed PROforma
Size	246	376
Depth	5	8
Plans	61	131
Decisions	21	57
Actions	105	188
Enquiries	59	0

number of plans in the transformed model is far larger than in the direct model. The reason is that every XOR/OR parallel component has been mapped to a plan. This explains also the difference in the depth of the direct and the transformed models. The difference in the number of decisions is also remarkable. The reason is that the modeler of the direct PROforma model used some enquiries as decisions, according to her experience and criteria. In contrast, there are not enquiries in the transformed PROforma model.

As mentioned before, we have manually compared the transformed model with the direct one, to check whether corresponding structures were equivalent. As an illustration, we analyze a decision in the transformed model and its corresponding decision in the direct one. Concretely, the clinical guideline recommends three different treatments for patients depending on their screening and cancer stage. This recommendation is modeled in BPMN with an XOR-gateway with three outgoing sequence flows, as shown in Fig. 1. Likewise, a decision with three candidates has been obtained in the transformed model. Table 4 lists these candidates, together with the argument and the recommendation rule for each candidate. Note that each candidate has a single argument, which corresponds to the condition of the BPMN sequence flow. Note also that the recommendation rule of all candidates states that the net support equals to one. This implies that a candidate will be chosen when the condition of its only argument is satisfied.

In the direct model, the corresponding decision has also three candidates but each candidate has multiple arguments, as shown in Table 5. Despite the differences in candidate details, the set of arguments and recommendation rules can be regarded as semantically equivalent. For instance, the recommendation rule for the second candidate of Table 5 requires that the net support is greater than or equal to one, which means that at least one of the conditions of the two arguments should be fulfilled. At the same time, the only argument of the second candidate in Table 4 contains a disjunction of roughly the same conditions. Together with the recommendation rule, which requires that the net support equals to one, this candidate will be selected exactly in the same situations.

We can draw several lessons from our experiments. The results obtained show that the implemented algorithms can successfully transform the BPMN specification of a realistic guideline into the PROforma language. This transformation is mostly done automatically, although a manual review of the resulting model is required in points where the degree of detail is greater than in the source (such as logical expressions). In this sense, we regard the transformation as semi-automatic. In general, a transformed model will have a higher number of elements (and a greater depth) than a manually developed one. However, in our view the models are always comparable. This means, we can draw a parallel between the corresponding components, and we can see that these components are semantically equivalent. Also as a consequence of the experiments, we have a clearer idea of how the implemented algorithms can be applied. We envisage an initial BPMN modeling performed mostly by clinicians, followed by the application of the algorithms and the manual revision of the resulting model by IT engineers. In the end, a joint review of the final model and its components can be made, if necessary using the information on the mappings to trace back to the originating components.

5 Conclusions

In this paper we introduce an approach that supports the refinement of clinical guidelines from an initial specification in a business process language to a detailed and executable implementation in one of the available CIG languages. In essence, our approach relies on a semi-automatic transformation from a BPMN specification of a

Table 4 Candidates, arguments and recommendation rules for decision *main_treatment_dec*, in the transformed PROforma model. HRFP stands for High Risk Factors Present

Candidate	Arguments	Rule
T_no_further_workup	(for) life_expectancy $\leq$ 5 and asymptomatic=true and HRFP=false	netsupport(main_treatment_decision, T_no_further_workup) = 1
SP_staging_workup	(for) life_expectancy>5 or symptomatic=true	netsupport(main_treatment_decision, SP_staging_workup) = 1
T_special_treatment _for_highrisk_patients	(for) life_expectancy $\leq$ 5 and asymptomatic=true and HRFP=true	netsupport(main_treatment_decision, T_special_treatment_for_highrisk _patients) = 1

Table 5 Candidates, arguments and recommendation rules for decision *treatment_decision*, in the direct PROforma model

Candidate	Arguments	Rule
No_treatment	(for) life_expectancy $\leq$ 5	netsupport(treatment_decision, No_Treatment) = 3
	(for) symptomatology="asymptomatic"	
	(for) not (bulky_cancer=true and (TNM="T3a" or TNM="T3b" or TNM="T4"))	
Treatment	(for) life_expectancy>5	netsupport(treatment_decision, Treatment) $\geq$ 1
	(for) symptomatology="symptomatic"	
Treatment_for _HighRisk	(for) life_expectancy $\leq$ 5	netsupport(treatment_decision, Treatment_for_HighRisk) = 3
	(for) symptomatology="asymptomatic"	
	(for) bulky_cancer=true and (TNM="T3a" or TNM="T3b" or TNM="T4")	

clinical guideline into an implementation in a CIG language. The importance of our transformation-based refinement approach lies in the fact that it can ultimately facilitate and speed up the development process of decision-support systems based on clinical guidelines. BPMN is a widely-adopted standard notation for business process modeling, able to support not only organizational processes but also clinical ones. BPMN can be easily understood by all stakeholders and thus has the potential to empower clinicians to address the guideline modeling task. This is crucial because the collaboration of clinical and IT staff has proven superior for this task (Patel et al. 1998). Accordingly, the combined use of BPMN and a CIG language, targeting clinicians and IT engineers, respectively, is a key feature of our approach.

In addition, in this paper we describe the algorithms that we have developed for the transformation of guideline models in BPMN into the PROforma language. The results obtained by applying the implemented algorithms to different guidelines show that a transformation from BPMN to PROforma, and hence the approach, is feasible. Moreover, the resulting models are of a reasonable quality, although a manual revision by IT engineers is always necessary due to the greater degree of detail of PROforma. One limitation is that the models resulting from the transformation are of greater structural complexity, when compared with models obtained in a fully manual way by IT engineers. Despite this, we hypothesize a higher degree of acceptance by clinicians, derived from a greater involvement in the initial BPMN modeling. Additionally, the use of BPMN by clinical experts can facilitate modifications and updates of the guideline model, which may be needed on a regular basis for certain diseases. These hypotheses have yet to be validated. A more general limitation is that a complete transformation might not be possible due to the different expressiveness of the source and target languages. It is therefore important to fully characterize the transformation algorithms developed in our approach, and to take into account these characteristics when applying the transformations.

Our solution is tailored to the features of BPDs representing clinical procedures. Thus, it considers subprocesses and specific process structures commonly found in clinical guidelines. To our knowledge, the only transformation approaches in the context of clinical guidelines are the works by González-Ferrer et al. (2013) and by Domínguez et al. (2010). González-Ferrer et al. tackle the transformation from XPDL to a HTN language, and Domínguez et al. implement Java modules from UML state diagrams. Therefore, none of them specifically deal with CIG languages.

An interesting aspect of our approach is that part of the algorithms can be reused to transform BPMN to other CIG languages. Only the last step, the generation of the target code, would have to be implemented. In this line, we have recently developed algorithms to transform BPMN guidelines to the SDA CIG language (Martínez-Salvador et al. 2015). Thus, from the same clinical guideline specification, we could obtain executable models in different CIG languages.

As future work, we plan to conduct experiments to assess the effectiveness of our approach with respect to our initial goal, which is to facilitate the encoding of clinical guidelines and simultaneously to involve clinicians more actively in the process. The setting for these experiments should be as realistic as possible, and compel clinicians and IT engineers to collaborate. Furthermore, we plan to incorporate Model-Driven Engineering techniques into our transformation algorithms. For this purpose, a logical continuation would be to define a model-driven transformation for each component identified in the source model.

Acknowledgments This research has been supported by Universitat Jaume I through Project P1·1B2013-15, and by the Spanish Ministry of Economy and Competitiveness and the EU FEDER programme through Project TIN2014-53749-C2-1-R.

References

Bae J, Bae H, Kang SH, Kim Y (2004) Automatic control of workflow processes using ECA rules. IEEE Trans Knowl Data Eng 16(8):1010–1023

Boxwala AA, Tu S, Peleg M, Zeng Q, Ogunyemi O, Greenes RA, Shortliffe EH, Patel VL (2001) Toward a representation format for sharable clinical guidelines. J Biomed Inform 34(3):157–169

Bury J, Hurt C, Aea Roy (2005) Lisa: a web-based decision-support system for trial management of childhood acute lymphoblastic leukaemia. Br J Haematol 129:746–754

COSSAC (2013a) Credo project. Interdisciplinary Research Collaboration in Cognitive Science and Systems Engineering (COSSAC). https://cossac.org/projects/credo. Accessed 28 July 2016

COSSAC (2013b) Tallis training. Interdisciplinary Research Collaboration in Cognitive Science and Systems Engineering. http://archive.cossac.org/tallis/index.html. Accessed 28 July 2016

Coulson A, Glasspool D, Fox J, Emery J (2001) Rags: a novel approach to computerised genetic risk assessment and decision support from pedigrees. Methods Inform 40:315–322

de Clercq PA, Blom JA, Korsten HH, Hasman A et al (2004) Approaches for creating computer-interpretable guidelines that facilitate decision support. Artif Intell Med 31(1):1–28

Domínguez E, Pérez B, Zapata M (2010) Towards a traceable clinical guidelines application. A model-driven approach. Methods Inf Med 49(6):571–580

Dugan L, Palmer N (2012) BPMN 2.0 handbook second edition: updated and expanded. Making a BPMN 2.0 model executable, Future Strategies Inc., Book Division, pp 71–91

Emery J, Walton R, Murphy M et al (2000) Computer support for interpreting family histories of breast and ovarian cancer in primary care: comparative study with simulated cases. Br Med J 321:28–32

Field MJ, Lohr KN (1990) Clinical practice guidelines: directions for a new program. The National Academies Press, Committee to Advise the Public Health Service on Clinical Practice Guidelines, Institute of Medicine

González-Ferrer A, Fernández-Olivares J, Castillo L (2013) From business process models to hierarchical task network planning domains. Knowl Eng Rev 28(2):175–193

Götz M, Roser S, Lautenbacher F, Bauer B (2009) Token analysis of graph-oriented process models. In: 13th IEEE enterprise distributed object computing conference workshops (EDOCW), pp 15–24

Grando MA, Glasspool D, Fox J (2012) A formal approach to the analysis of clinical computer-interpretable guideline modeling languages. Artif Intell Med 54(1):1–13

Hashemian N, Abidi SSR (2012) Modeling clinical workflows using business process modeling notation. In: 25th international symposium on computer-based medical systems (CBMS), pp 1–4

Kiepuszewski B, Maria ter Hofstede AH, Bussler CJ (2000) On structured workflow modelling. LNCS 1789. Springer, Heidelberg

Kirchner K, Malessa C, Scheuerlein H, Settmacher U (2014) Experience from collaborative modeling of clinical pathways. In: Hess M, Schlieter H (eds) Modellierung im Gesundheitswesen: Tagungsband des Workshops im Rahmen der Modellierung, p 13

Marcos M, Balser M, ten Teije A, van Harmelen F, Duelli C (2003) Experiences in the formalisation and verification of medical protocols. Artificial intelligence in medicine. Springer, Heidelberg

Martínez-Salvador B, Marcos M, Sánchez A (2014) An algorithm for guideline transformation: from BPMN to PROforma. Knowledge representation for health care. Springer, Heidelberg

Martínez-Salvador B, Marcos M, Riaño D (2015) An algorithm for guideline transformation: from BPMN to SDA. Procedia Comput Sci 63:244–251

Mendling J, Lassen KB, Zdun U (2008) On the transformation of control flow between block-oriented and graph-oriented process modelling languages. Int J Bus Process Integr Manag 3(2):96–108

Mendling J, Reijers H, van der Aalst W (2010) Seven process modeling guidelines (7 pgm). Inf Softw Technol 52:127–136

Mohler J, Amstrong A, Bahnson R, Boston B, Busby J, D'Amico A, Eastham J, Enke C, Farrington T, Higano C, Horwitz E, Kantoff P, Kawachi M, Kuette M, Lee R, MacVicar G, Malcolm A, Miller D, Plimack E, Pow-Sang J, Mr Roach, Rohren E, Rosenfeld S, Srinivas S, Strope S, Tward J, Twardowski P, Walsh P, Ho M, Sheadm D (2012) Prostate cancer, version 3.2012: featured updates to the NCCN guidelines. J Natl Compr Cancer Netw 10(9):1081–1087

OMG (2011) Busines process model and notation (BPMN) version 2.0. http://www.omg.org/spec/BPMN/2.0. Accessed 28 July 2016

Ouyang C, Dumas M, Aalst WM, Hofstede AHT, Mendling J (2009) From business process models to process-oriented software systems. ACM Transactions on Software Engineering and Methodology (TOSEM) 19(1):2

Parra C, Jódar-Sánchez F, Jiménez-Hernández MD, Vigil E, Palomino-García A, Moniche-Álvarez F, De la Torre-Laviana FJ, Bonachela P, Fernández FJ, Cayuela-Domínguez A et al (2012) Development, implementation, and evaluation of a telemedicine service for the treatment of acute stroke patients: telestroke. Interact J Med Res 1(2)

Patel VL, Allen VG, Arocha JF, Shortliffe EH (1998) Representing clinical guidelines in GLIF individual and collaborative expertise. J Am Med Inform Assoc 5(5):467–483

Peleg M, Tu S, Bury J, Ciccarese P, Fox J, Greenes RA, Hall R, Johnson PD, Jones N, Kumar A et al (2003) Comparing computer-interpretable guideline models: a case-study approach. J Am Med Inform Assoc 10(1):52–68

Recker J (2010) Opportunities and constraints: the current struggle with bpmn. Bus Process Manag J 16(1):181–201

Rojo MG, Rolón E, Calahorra L, García F, Sánchez RP, Ruiz F, Ballester N, Armenteros M, Rodríguez T, Espartero RM et al (2008) Implementation of the business process modelling notation (BPMN) in the modelling of anatomic pathology processes. Diagn Pathol 3(Suppl 1):S22

Rolón E, García F, Ruiz F, Piattini M, Calahorra L, García M, Martin R (2008) Process modeling of the health sector using bpmn: a case study. In: Proceedings of the first international conference on health informatics (HEALTHINF), diagnostic pathology, vol 2, pp 173–178

Scheuerlein H, Rauchfuss F, Dittmar Y, Molle R, Lehmann T, Pienkos N, Settmacher U (2012) New methods for clinical pathways – business process modeling notation (BPMN) and tangible business process modeling (t. BPM). Langenbeck's. Arch Surg 397(5):755–761

Siegel R, Naishadham D, Jeme A (2013) Cancer statistics. Cancer J Clin 63(1):11–30

Sutton DR, Fox J (2003) The syntax and semantics of the proforma guideline modeling language. J Am Med Inform Assoc 10 (5):433–443

Svagård I, Farshchian BA (2009) Using business process modelling to model integrated care processes: experiences from a European project. Distributed computing, artificial intelligence, bioinformatics, soft computing, and ambient assisted living. Springer, Heidelberg

Van der Aalst WM, Ter Hofstede AH, Kiepuszewski B, Barros AP (2003) Workflow patterns. Distrib Parallel Databases 14(1):5–51

zur Muehlen M, Recker J (2008) How much language is enough? Theoretical and practical use of the business process modeling notation. In: 20th international conference on advanced information systems engineering, LNCS. Springer, Heidelberg