

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

Begoña Martínez-Salvador and Mar Marcos

Cite this article as:

Martínez-Salvador, B. & Marcos, M. Bus Inf Syst Eng (2016) 58: 355.

doi:10.1007/s12599-016-0443-3

NOTICE: This is the author's version of a work that was accepted for publication in Business & Information Systems Engineering Journal. Changes resulting from the publishing process, such as peer review, editing, corrections, structural formatting, and other quality control mechanisms may not be reflected in this document. Changes may have been made to this work since it was submitted for publication. A definitive version was subsequently published in Business & Information Systems Engineering (BISE) journal Volume 58, Issue 5, October 2016, pages 355-366. DOI information 10.1007/s12599-016-0443-3. **The final publication is available at Springer via <http://dx.doi.org/10.1007/s12599-016-0443-3>**

Corresponding author: Begoña Martínez-Salvador, Department of Computer Engineering and Science Universitat Jaume I, Castellón, Spain. E-mail: begona.martinez@uji.es

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

Begoña Martínez-Salvador, Ph.D. ·
Mar Marcos, Ph.D.

Received: date / Accepted: date

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. Our goal is to facilitate the encoding of clinical guidelines in CIG languages, while increasing the involvement of clinicians. To achieve this we propose 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. Our approach relies on the use of the Business Process Modelling 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 our approach, in this work we have implemented algorithms to transform a BPMN specification of clinical processes into the PROforma CIG language, and we have successfully applied them to several clinical guidelines.

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

1 Introduction

Clinical guidelines are defined as “systematically developed statements to assist practitioners and patient decisions about appropriate health care for specific circumstances” [Field & 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

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

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 translate the clinical guideline to 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 reason, in the Medical Informatics area, several languages for modelling 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. Although there have been some attempts, 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 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 analyse 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 Modelling 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 modelling step plus a semi-automatic transformation step. An important advantage of the initial modelling 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 modelling language, such as BPMN. Because the last BPMN specification [OMG, 2011] provides some execution semantics in terms of BPEL, in general BPEL is mistaken as an executable expression of BPMN. However, the full equivalence of BPMN cannot be expressed in BPEL [Dugan & 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 modelling [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 modelling of clini-

cal 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 modelling 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 and computer science 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 Section 4 some experimental results with a prostate cancer guideline are presented. Finally Section 5 concludes and outlines some future work.

2 MATERIAL AND METHODS

2.1 BPMN

The Object Management Group (OMG) has developed the BPMN notation, that 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 modelling processes for patients with a particular condition [Rojo et al., 2008], [Rolón et al., 2008], [Parra et al., 2012]. Others use BPMN for modelling clinical pathways [Svagård & Farshchian, 2009], [Scheuerlein et al., 2012], [Hashemian & Abidi, 2012], [Kirchner et al., 2014]. Most of the works agree in 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 modelling 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 modelling 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 et al. [zur Muehlen & 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 [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 Section 4).

BPMN flow objects are the main elements for defining the behaviour 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 behaviour. 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 modelled 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 any 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 out from split gateways optionally define a condition expression to be evaluated before deciding whether or not follow that flow. In the case of split

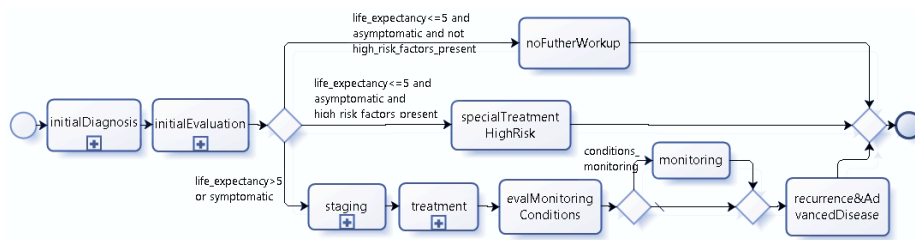


Fig. 1 Main clinical process in BPMN for the diagnosis and treatment of prostate cancer. Events are depicted as circles. Rounded boxes are tasks, or sub-processes if they have the mark +. Gateways are depicted as diamonds and sequence flows as arrows.

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 & Fox, 2003]. The Tallis implementation [COSSAC, 2013b] provides a Composer for authoring CIG-based decision support systems that might 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 ask 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 some 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 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*).

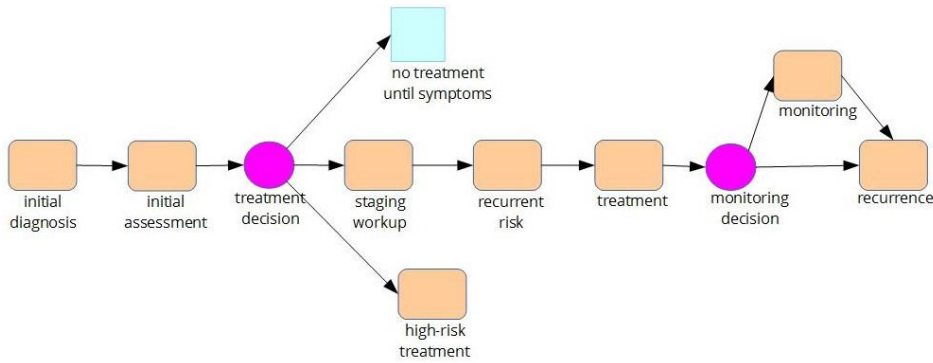


Fig. 2 PROforma graphical notation of the main clinical process for the diagnosis and treatment of prostate cancer. Plans are depicted as rounded boxes, actions as squares and decisions as circles. Arcs between tasks represent scheduling constraints.

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 have several sources and each source is based on a *data definition*.

Process descriptions are modelled 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 Figure 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 modelling 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 *structure-identification strategy*.

This strategy consists in, starting from certain structures of interest in the target language, 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, 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 Section 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

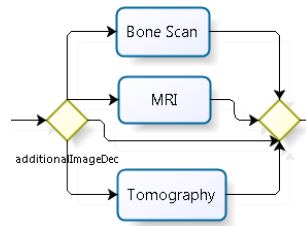


Fig. 3 XOR-parallel component with an arc that directly connects the split with the join gateway, modelling the option “no imaging for some patients”.

guidelines are formulated in natural language, non-structuredness is not 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. [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. Figure 1). This is in line with one of the seven process modelling guidelines by Mendling et al. [Mendling et al., 2010].

Additionally, clinical guidelines may contain recommendations that can be modelled as iterative processes. We have considered the usage of loops in tasks or sub-processes for modelling 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 modelled with a sequence flow that directly connects the split with the join XOR-gateway, as Figure 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 an 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.

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 between the split and join gateway of an XOR or OR-parallel component there can be any type of elements including other components, 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 Section 2.2). In general, the successor nodes of the split XOR-gateway will give rise to the candidates, although the transformation also works with XOR-parallel components like the one in Figure 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, like the loop condition, that will be mapped to a PROforma attribute. 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 API¹ for manipulating XML data.

¹ <http://www.jdom.org/docs/apidocs/org/jdom2/input/SAXBuilder.html> (last access:13-06-13)

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 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 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 modelled them using an AND-gateway with an arc for each inner process. On the other hand, to mimic the behaviour of a sequential ad-hoc sub-process, each possible sequential arrangement of the inner processes has been modelled 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 Figure 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 Algorithm 2 and Algorithm 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 losing 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.

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.

Algorithm 1: ComponentIdentificationAlg

Input: Graph, Weights, GatewayNodes
Output: Trivial graph, SequenceComponents, ParallelComponents
allSequencesSearch(start node, Graph, SequenceComponents) ;
while (*non trivial graph*) **do**
parallelComponent \leftarrow parallelComponentSearch(Graph, Weights, GatewayNodes,
ParallelComponents) ;
if (*ParallelComponent done*) **then**
add(ParallelComponents, parallelComponent) ;
parallelComponentNode \leftarrow 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 \leftarrow 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;

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.

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  $\leftarrow$  replace (Graph, parallelComponent) ;
end
return Graph, parallelComponentNode, ParallelComponents

```

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 modelled this guideline both in BPMN (Figure 1) and in PROforma (Figure 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 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 the execution of the transformed model produces the same results that would be obtained by applying the text version of the guideline, for a series of test cases. 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 modelled in PROforma –which we refer to as *direct model*– to determine the equivalence of corresponding structures.

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 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

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 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

Section 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 modeller 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 number of plans in the transformed model is far bigger 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 modeller 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 analyse a decision in the transformed model and its corresponding

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

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 modelled in BPMN with an XOR-gateway with three outgoing sequence flows, as shown in Figure 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 level 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. That is, 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 more clear idea of how the implemented algorithms can be applied. We envisage an initial BPMN modelling 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 the originating components.

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

candidate	arguments	rule
No_treatment	(for) $\text{life_expectancy} \leq 5$	$\text{netsupport}(\text{treatment_decision}, \text{No_Treatment}) = 3$
	(for) $\text{symptomatology} = \text{"asymptomatic"}$	
	(for) $\text{not}(\text{bulky_cancer} = \text{true} \text{ and } (\text{TNM} = \text{"T3a"} \text{ or } \text{TNM} = \text{"T3b"} \text{ or } \text{TNM} = \text{"T4"}))$	
Treatment	(for) $\text{life_expectancy} > 5$	$\text{netsupport}(\text{treatment_decision}, \text{Treatment}) \geq 1$
	(for) $\text{symptomatology} = \text{"symptomatic"}$	
Treatment_for_HighRisk	(for) $\text{life_expectancy} \leq 5$	$\text{netsupport}(\text{treatment_decision}, \text{Treatment_for_HighRisk}) = 3$
	(for) $\text{symptomatology} = \text{"asymptomatic"}$	
	(for) $\text{bulky_cancer} = \text{true} \text{ and } (\text{TNM} = \text{"T3a"} \text{ or } \text{TNM} = \text{"T3b"} \text{ or } \text{TNM} = \text{"T4"})$	

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 clinical guideline into an implementation in a CIG language. The importance of our transformation-based refinement approach lies in 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 modelling, 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 modelling 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 level of detail of PROforma. One limitation is that the models resulting from the transformation are of greater structural complexity, when compared with models obtained fully manually by IT engineers. Despite this, we hypothesize a higher degree of acceptance by clinicians, derived from a greater involvement in the initial BPMN modelling. 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 characterize fully 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 sub-processes 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. [González-Ferrer et al., 2013] and by Domínguez et al. [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 the joint participation of clinicians and IT engineers. 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.

Acknowledgements 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 et al., 2004] Bae, Joonsoo, Hyerim Bae, Suk-Ho Kang, & Yeongho Kim 2004. Automatic control of workflow processes using ECA rules. *Knowledge and Data Engineering, IEEE Transactions on*, 16(8):1010–1023.
- [Boxwala et al., 2001] Boxwala, Aziz A, Samson Tu, Mor Peleg, Qing Zeng, Omolola Ogunyemi, Robert A Greenes, Edward H Shortliffe, & Vimla L Patel 2001. Toward a representation format for sharable clinical guidelines. *Journal of Biomedical Informatics*, 34(3):157–169.
- [Bury et al., 2005] Bury, J., C. Hurt, & A. et al. Roy 2005. LISA: a web-based decision-support system for trial management of childhood acute lymphoblastic leukaemia. *British Journal of Haematology*, 129:746–754.
- [COSSAC, 2013a] COSSAC 2013a. CREDO Project. <http://www.cossac.org/projects/credo>. Interdisciplinary Research Collaboration in Cognitive Science & Systems Engineering (COSSAC).
- [COSSAC, 2013b] COSSAC 2013b. Tallis Training. <http://archive.cossac.org/tallis/index.html>. Interdisciplinary Research Collaboration in Cognitive Science & Systems Engineering.
- [Coulson et al., 2001] Coulson, A., DW. Glasspool, J. Fox, & J. Emery 2001. RAGS: a novel approach to computerised genetic risk assessment and decision support from pedigrees. *Methods of Infor*, 40:315–322.

- [de Clercq et al., 2004] de Clercq, Paul A, Johannes A Blom, Hendrikus HM Korsten, Arie Hasman, et al. 2004. Approaches for creating computer-interpretable guidelines that facilitate decision support. *Artificial Intelligence in Medicine*, 31(1):1–28.
- [Domínguez et al., 2010] Domínguez, E., B. Pérez, & M. Zapata 2010. Towards a traceable clinical guidelines application. A model-driven approach. *Methods of Information in Medicine*, 49(6):571–580.
- [Dugan & Palmer, 2012] Dugan, Lloyd, & Nathaniel Palmer 2012. *BPMN 2.0 Handbook Second Edition: Updated and Expanded*, chapter Making a BPMN 2.0 Model Executable, pages 71–91. Future Strategies Inc., Book Division.
- [Emery et al., 2000] Emery, J., R. Walton, M. Murphy, & et al. 2000. Computer support for interpreting family histories of breast and ovarian cancer in primary care: comparative study with simulated cases. *British Medical Journal*, 321:28–32.
- [Field & Lohr, 1990] Field, Marilyn J., & Kathleen N. Lohr 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 et al., 2013] González-Ferrer, Arturo, Juan Fernández-Olivares, & Luis Castillo 2013. From Business Process Models to Hierarchical Task Network Planning Domains. *Knowl. Eng. Rev.*, 28(2):175–193.
- [Götz et al., 2009] Götz, M, Stephan Roser, Florian Lautenbacher, & Bernhard Bauer 2009. Token analysis of graph-oriented process models. In *Enterprise Distributed Object Computing Conference Workshops, 2009. EDOCW 2009. 13th*, pages 15–24. IEEE, IE.
- [Grando et al., 2012] Grando, M Adela, David Glasspool, & John Fox 2012. A formal approach to the analysis of clinical computer-interpretable guideline modeling languages. *Artificial Intelligence in Medicine*, 54(1):1–13.
- [Hashemian & Abidi, 2012] Hashemian, Nima, & Syed Sibte Raza Abidi 2012. Modeling clinical workflows using business process modeling notation. In *Computer-Based Medical Systems (CBMS), 2012 25th International Symposium on*, pages 1–4. IEEE, IEEE.
- [Kiepuszewski et al., 2000] Kiepuszewski, B., A. H. Maria ter Hofstede, & C. J. Bussler 2000. On Structured Workflow Modelling. In *Lecture Notes in Computer Science*, number 1789, pages 431–445. Springer-Verlag, Berlin Heidelberg. (CAiSE 2000).
- [Kirchner et al., 2014] Kirchner, Kathrin, Christina Malessa, Hubert Scheuerlein, & Utz Settmacher 2014. Experience from collaborative modeling of clinical pathways. In Hess, M., & H. Schlieter (eds), *Modellierung im Gesundheitswesen: Tagungsband des Workshops im Rahmen der Modellierung*, page 13.
- [Marcos et al., 2003] Marcos, Mar, Michael Balsler, Annette ten Teije, Frank van Harmelen, & Christoph Duelli 2003. Experiences in the formalisation and verification of medical protocols. In *Artificial Intelligence in Medicine*, pages 132–141. Springer.
- [Martínez-Salvador et al., 2015] Martínez-Salvador, Begoña, Mar Marcos, & David Riaño 2015. An Algorithm for Guideline Transformation: From BPMN to SDA. *Procedia Computer Science*, 63:244–251.
- [Martínez-Salvador et al., 2014] Martínez-Salvador, Begoña, Mar Marcos, & Anderson Sánchez 2014. An algorithm for guideline transformation: from BPMN to PROforma. In *Knowledge Representation for Health Care*, pages 121–132. Springer.
- [Mendling et al., 2008] Mendling, Jan, Kristian Bisgaard Lassen, & Uwe Zdun 2008. On the transformation of control flow between block-oriented and graph-oriented process modelling languages. *International Journal of Business Process Integration and Management*, 3(2):96–108.
- [Mendling et al., 2010] Mendling, J., H.A. Reijers, & W.M.P. van der Aalst 2010. Seven process modeling guidelines (7PGM). *Information and Software Technology*, 52:127–136.
- [Mohler et al., 2012] Mohler, J.L., A.J. Armstrong, R.R. Bahnson, B. Boston, J.E Busby, A.V. D’Amico, J.A. Eastham, C.A. Enke, T. Farrington, C.S. Higano, EM Horwitz, PW Kantoff, MH Kawachi, I M Kvette, RJ Lee, GR MacVicar, AW Malcolm, D Miller, ER Plimack, JM Pow-Sang, M 3rd Roach, E Rohren, S Rosenfeld, S Srinivas, SA Strobe, J Tward, P Twardowski, PC Walsh, M Ho, & DA. Sheadm 2012. Prostate cancer, Version 3.2012: featured updates to the NCCN Guidelines. *Journal of the National Comprehensive Cancer Network*, 10(9):1081–1087.
- [OMG, 2011] OMG 2011. *Business Process Model and Notation (BPMN) Version 2.0*.
- [Ouyang et al., 2009] Ouyang, Chun, Marlon Dumas, Wil MP Aalst, Arthur HM Ter Hofstede, & Jan Mendling 2009. From business process models to process-oriented software systems. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 19(1):2.

- [Parra et al., 2012] Parra, Carlos, Francisco Jódar-Sánchez, M Dolores Jiménez-Hernández, Eduardo Vigil, Alfredo Palomino-García, Francisco Moniche-Álvarez, Francisco Javier De la Torre-Laviana, Patricia Bonachela, Francisco José Fernández, Aurelio Cayuela-Domínguez, et al. 2012. Development, implementation, and evaluation of a telemedicine service for the treatment of acute stroke patients: teleStroke. *Interactive Journal of Medical Research*, 1(2).
- [Patel et al., 1998] Patel, Vimla L, Vanessa G Allen, José F Arocha, & Edward H Shortliffe 1998. Representing Clinical Guidelines in GLIF Individual and Collaborative Expertise. *Journal of the American Medical Informatics Association*, 5(5):467–483.
- [Peleg et al., 2003] Peleg, Mor, Samson Tu, Jonathan Bury, Paolo Ciccarese, John Fox, Robert A Greenes, Richard Hall, Peter D Johnson, Neill Jones, Anand Kumar, et al. 2003. Comparing computer-interpretable guideline models: a case-study approach. *Journal of the American Medical Informatics Association*, 10(1):52–68.
- [Recker, 2010] Recker, Jan 2010. Opportunities and constraints: the current struggle with BPMN. *Business Process Management Journal*, 16(1):181–201.
- [Rojo et al., 2008] Rojo, Marcial García, Elvira Rolón, Luis Calahorra, Félix García, Rosario Paloma Sánchez, Francisco Ruiz, Nieves Ballester, María Armenteros, Teresa Rodríguez, Rafael Martín Espartero, et al. 2008. Implementation of the Business Process Modelling Notation (BPMN) in the modelling of anatomic pathology processes. *Diagnostic Pathology*, 3(Suppl 1):S22.
- [Rolón et al., 2008] Rolón, Elvira, Félix García, Francisco Ruiz, Mario Piattini, Luis Calahorra, Marcial García, & Rafael Martín 2008. Process modeling of the health sector using BPMN: A case study. In *Proceedings of the First International Conference on Health Informatics, HEALTHINF 2008*, volume 2, pages 173–178. *Diagnostic Pathology*.
- [Scheuerlein et al., 2012] Scheuerlein, Hubert, Falk Rauchfuss, Yves Dittmar, Rüdiger Molle, Torsten Lehmann, Nicole Pienkos, & Utz Settmacher 2012. New methods for clinical pathways – Business Process Modeling Notation (BPMN) and Tangible Business Process Modeling (t. BPM). *Langenbeck’s Archives of Surgery*, 397(5):755–761.
- [Siegel et al., 2013] Siegel, R., D. Naishadham, & A. Jem 2013. Cancer Statistics. *A Cancer Journal for Clinicians*, 63(1):11–30.
- [Sutton & Fox, 2003] Sutton, David R, & John Fox 2003. The syntax and semantics of the PROforma guideline modeling language. *Journal of the American Medical Informatics Association*, 10(5):433–443.
- [Svagård & Farshchian, 2009] Svagård, Ingrid, & Babak A Farshchian 2009. Using Business Process Modelling to Model Integrated Care Processes: Experiences from a European Project. In *Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living*, pages 922–925. Springer.
- [van der Aalst et al., 2003] van der Aalst, Wil MP, Arthur HM Ter Hofstede, Bartek Kiepuszewski, & Alistair P Barros 2003. Workflow patterns. *Distributed and Parallel Databases*, 14(1):5–51.
- [zur Muehlen & Recker, 2008] zur Muehlen, Michael, & Jan Recker 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-Verlag.