Representation of Clinical Practice Guideline Components in OWL

Tiago Oliveira[1], Paulo Novais[1] and José Neves[1]

[1]CCTC/DI, University of Minho, Braga, Portugal

[1]{toliveira,pjon,jneves}@di.uminho.pt

Abstract. The main purpose to attain with the advent of clinical decision support systems is either to improve the quality of patient care or to reduce the occurrence of clinical malpractice, such as medical errors and defensive medicine. It is therefore necessary a machine-readable support to integrate the recommendations of Clinical Practice Guidelines in such systems. CompGuide is a Computer-Interpretable Guideline model developed under Ontology Web Language that offers support for administrative information concerning a guideline, workflow procedures, and the definition of clinical and temporal constraints. When compared to other models of the same type, besides having a comprehensive task network model, it introduces new temporal representations and the possibility of reusing pre-existing knowledge and integrating it in a guideline.

Keywords: Clinical Practice Guidelines, Ontology, OWL, Decision Support Systems

1 Introduction

Among the healthcare community the occurrence of medical errors and defensive medicine are of uppermost concern [1][2]. Medical errors refer to mistakes during the clinical process that may lead to adverse events, i.e., adjustments in a patient's health condition for the worst [1]. It includes errors of execution, treatment and planning, and their incidence rates, although not very high, are associated with increased spending and loss of life quality for both physicians and patients [2]. To avoid these situations, healthcare professionals often adopt another type of harmful behavior, namely defensive medicine. Indeed, defensive medicine consists in avoiding the treatment of difficult clinical cases to prevent possible lawsuits or ordering additional complementary means of diagnostic, motivated by the sense of self-preservation. This behavior is also motivated by the overreliance on technological means for diagnostic purposes, which in turn are responsible for rising healthcare costs [3]. If one wants to reduce the impact of medical malpractice, it is necessary to standardize healthcare delivery and provide adequate evidence-based recommendations for clinical encounters [4]. Clinical Practice Guidelines (CPGs) are the current medium of choice to disseminate evidence-based medicine.

According to the definition of the Institute of Medicine (IOM) of the United States (US), CPGs are methodically advanced statements that contain recommendations for healthcare professionals and patients about appropriate medical procedures in specific clinical circumstances [5], being regarded by healthcare professionals as vehicles that may integrate the most current indications into patient management [6]. However, the current format of CPGs presents some weaknesses. They are available as very long documents that are difficult to consult, since only a small part of them refer truly to clinical recommendations. Moreover, there are some issues concerning their ambiguity [7], namely the misunderstanding of medical terms (semantic ambiguity); conflicting instructions (pragmatic ambiguity); and the incorrect structure of statements (syntactic ambiguity). Additionally, some forms of vagueness may occur in the text, mainly due to the use of temporal terms (e.g. always, sometimes), probabilistic terms (e.g. probable, unlikely) and quantitative terms (e.g., many, few). A structured format for CPGs that is, at the same time, machine-readable, would help to solve these issues by providing an adequate support for guideline dissemination and deployment, at the moment of healthcare delivery [8].

This work presents a representation model for CPGs developed in Ontology Web Language (OWL) capable of accommodating guidelines from any category (e.g., diagnosis, evaluation, management and treatment) and medical sphere (e.g., family practice, pediatrics, and cardiology). As for the organization of this article, it is presented in section two the fundamentals about OWL along with some observations concerning the advantages of choosing this knowledge representation formalism over a traditional one, like relational databases. The model, which was named CompGuide, is presented in section three with the different requirements that were taken into consideration during the development phase. Section four presents a discussion about the advantages of the model in comparison with the existing ones, and provides some clarifications as well as future directions for this research.

2 Advantages of Ontology Web Language

The OWL Web Ontology Language is a standard developed by the World Wide Web Consortium (W3C), being its current version OWL 2 [9]. OWL is designed to be used by applications that need to process the content of information rather than just presenting information to humans. This formalism facilitates machine interpretability and is built upon other technologies such as XML, RDF and RDF-schema. OWL is composed of three sublanguages: OWL Lite, OWL DL and OWL Full. The sublanguage used for this work was OWL DL and it is named in this way due to its correspondence with description logics. An ontology is used to describe the concepts in a domain as well as the relationships that hold among them. To accomplish this task, OWL ontologies use three basic components:

- *Classes*, i.e., sets that contain individuals described using formal (mathematical) descriptions that state precisely the requirements for class membership;
- *Individuals*, i.e., objects of the domain and instances of classes; and

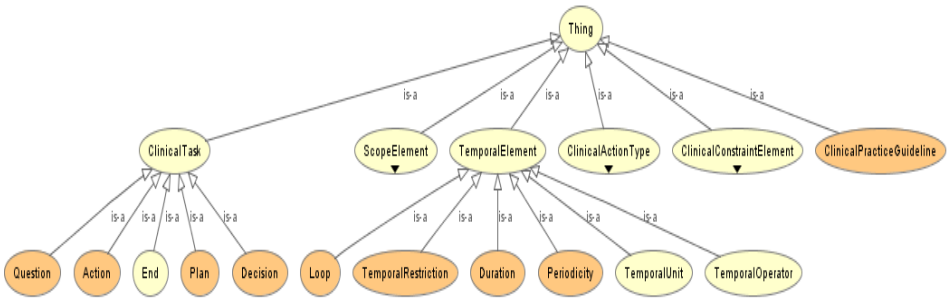

Fig. 1. Diagram of the main primitive classes in the Computer-Interpretable Guideline model.

- *Properties*, i.e., binary relations on individuals that may be used to link two individuals (object properties) or an individual to a data element (data properties).

The advantages of OWL reside in the manner a system uses the information. Machines do not grasp yet human language and, occasionally, there is content that escapes their understanding. For instance, a human being may comprehend that in some situations there are words that are unquestionably related, although not being their replacements. A machine does not recognize these relationships, but semantics are essential. Indeed, the idea behind OWL is to provide a machine with a semantic context; the advantage relies on the creation of a better management of the information and its descriptions. If the system is internal to an organization, there is no need to use OWL. However, if it is something that must be released into the world, OWL will probably be a better choice in the long term.

In OWL, semantic data is assembled into a graph database, that is, unlike the more common relational and hierarchical databases, built around nodes and tables. The relationships in OWL assume a greater importance and are the carriers of the semantic content of individuals. Moreover, it is possible to describe or restrain class membership using these relations, and thus delimit their scope in an accurate way. In relational databases this would be a hard task to perform. In fact, there is several software engines developed to reason about the semantic content of ontologies, which check the integrity of the constraints posed on individuals in order to assert if they belong or not to a certain class. Examples of such engines are Pellet, FaCT++ and HermiT, which are available as plugins for Protégé, the ontology editor and knowledge acquisition system used in this work. The reasoned used in this work was FaCT++. As the objective is the development of a standard machine-readable representation of CPGs, OWL appears to be the best formalism to use.

3 CompGuide Ontology

There are essentially two ways of developing Computer-Interpretable Guidelines (CIGs), namely by consulting domain experts in order to get the representation primitives, or by researching different CPGs and determine the information needed for clinical recommendations. The method followed in this work was a hybrid one, in the

sense that it includes opinions from healthcare professionals and the observation of guidelines collected from the National Guideline Clearinghouse (NGC)[1]. The guideline model was developed with the objective of fulfilling the requirements, i.e., representation of administrative information, construction of workflow procedures, and the definition of temporal and clinical constraints.

The key primitive classes of the model are depicted in Fig. 1, and will be described in detail in the following subsections.

3.1 Representation of Administrative Information

As it may be seen in Fig. 1, a CPG is represented as an instance of the *ClinicalPracticeGuideline* class. To keep track of different guideline versions and to provide rigorous descriptions of guideline content and objectives, the individuals of this class have a set of data properties that denote administrative information.

OWL has built-in data types that allow the expression of simple text, numeric values and dates. As such the *string* and *date-time* properties defined for administrative purposes were *Authorship*, *guidelineName*, *guidelineDescription*, *DateOfCreation*, *DateOfLastUpdate,* and *VersionNumber*. There are also additional properties that specify in which conditions and to whom the CPG should be applied, such as *ClinicalSpecialty*, *GuidelineCategory*, *intendedUsers,* and *targetPopulation*.

3.2 Construction of Workflow Procedures

CPGs are, essentially, clinical recommendations that are usually presented as series of tasks that must be performed during clinical encounters, and/or disease management processes. To represent these tasks, CompGuide proposes three main primitive classes, defined under *ClinicalTask*, in terms of *Plan*, *Action*, *Decision* and *Question*. The tasks of an individual from *ClinicalPracticeGuidelin*e are all contained in an individual *Plan*, to which it is linked through the *hasPlan* object property, as it may be seen in Fig. 2. This is an extract of the Standards of Medical Care in Diabetes guideline from the American Diabetes Association, obtained from the NGC. A *Plan* contains any number of instances belonging to other tasks, including other *Plans*, and it is connected to its first task through the *hasFirstTask* property. In turn, this task is linked to the next task in the workflow by the *nextTask* property, and so on. This assures the definition of a sequence of tasks in a manner similar to a linked list, as it is shown in Fig. 2.

The remaining task classes represent different types of activities. Starting with the *Action* class, it stands for a step performed by a healthcare agent that includes clinical procedures, clinical exams, and medication or non-medication recommendations. The *hasClinicalActionType* object property connects an *Action* to different action types defined in *ClinicalActionType*, which describe each one with appropriate data properties.

[1] http://guideline.gov/browse/by-topic.aspx

To express decision moments in the workflow, there is a *Decision* class. The use of this class entails a bifurcation in the clinical workflow and a choice between two or more options. The association of a *Decision* with options and rules is done through object properties that connect them to instances from the *ClinicalConstraintElement* subclasses. The next task in the clinical workflow is selected according to the outcome of the *Decision*. As so, the connection between these tasks is done using the *alternativeTasks* property. This assures that a task is executed instead of another as the result of an inference process guided by trigger conditions.

On the other hand, there may be cases when some tasks must be executed simultaneously, like the procedures of a treatment plan that act synergistically to produce a certain result. For these cases, CompGuide provides the *parallelTasks* property.

The *Question* class is used to obtain information about a patient's health condition, more specifically about the clinical parameters necessary to follow the guideline. In order to fulfill this requirement, there are data properties created to specify the name of the parameter to be obtained and the units in which it should be expressed. Such properties are *string* data types named *Parameter* and *Unit*, respectively.

The *End* class is used to signal the termination of the execution thread that is being followed and to indicate that the guideline reached its ultimate point.

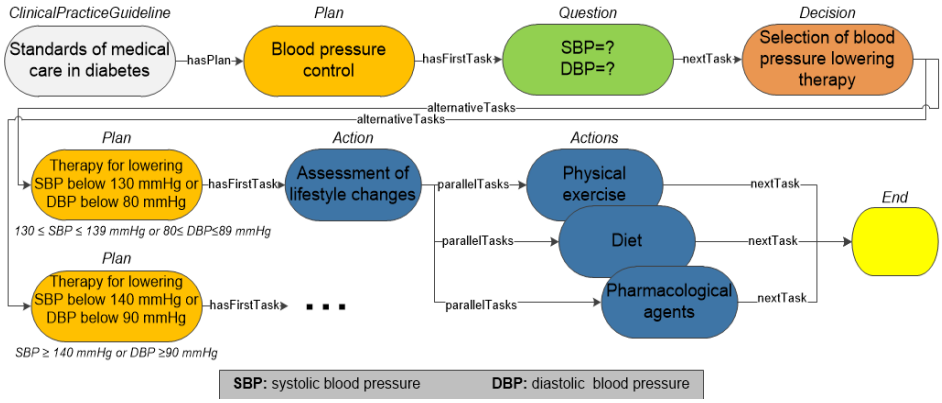

Fig. 2. Excerpt of the Standards of Medical Care in Diabetes clinical guideline from the American Diabetes Association, and represented according to the CompGuide model.

3.3 Definition of Temporal Constraints

The weight of time in clinical observations is paramount [10]. When assessing a patient state, a healthcare professional must take into consideration for how long the patient is manifesting his/her symptoms and try to fit this awareness in the one he/she already has about possible causes and solutions. The recommendations of CPGs contain specifications about their temporal execution, namely intended duration, number of repetitions and cycles.

To represent all the temporal constraints, CompGuide provides the *TemporalElement* class. This class includes two main temporal constructs, *Duration* and *Loop*. The

Duration class specifies how long a task should last, and is defined exclusively for *Plans* and *Actions*. It has a *double* data type property called *DurationValue,* where a value for the intended duration of either *Actions* or *Plans* is provided. The *TemporalUnit* class, also defined under *TemporalElement*, covers individuals that represent the different time units in which the *Duration* may be expressed, namely *second*, *minute*, *hour*, *day*, *week*, *month* and *year*. In the *Loop* class it is possible to define cycles for the executions of certain tasks (*Plans* and *Actions*). Each instance of *Loop* has a data property called *RepetitionValue,* which is an *integer* that expresses the number of repetitions a group of tasks is exposed to. Moreover, each instance also has a *hasPeriodicity* object property that connects it to individuals from the *Periodicity* class (another subclass of *TemporalElement),* which devises some constructs, namely the *hasTemporalUnit* object property and the *PeriodicityValue* data property, to define the regular intervals at which the task is repeated.

Another feature of the temporal properties is the possibility to define temporal restrictions in clinical constraints. For this purpose one associates a *TemporalRestriction* and a *TemporalOperator* to clinical conditions that must be met for a task to be executed. The temporal operators are based on the theory for quality checking of clinical guidelines by Peter Lucas [11], and include the following individuals:

- *Somewhere in the past*, i.e., the condition manifested at some point in the past;
- *Always in the past*, i.e., the condition was expressed during a time interval in the past; and
- *Currently*, i.e., the condition manifested during the medical observations.

The *TemporalRestriction* bounds the *TemporalOperator* to a defined period of time. It possesses a *double* data property labeled as *temporalRestrictionValue* and *hasTemporalUnit* object property. For instance, if an *Action* requires the verification if a patient has been doing is medication for the last 3 months, then *TemporalOperator* is set to *always in the past*, *temporalRestrictionValue* is set to *3* and finally *TemporalUnit* is set to *month*.

3.4 Definition of Clinical Constraints

As it was mentioned previously, a *Decision* implies a choice between two or more options. The association of individuals from *Option* (Fig. 3) to *Decision* is done by the *hasOption* property. The number of times this property is used in a *Decision* is equal to the number of options the task denotes. Each *Option* has a *Parameter* and a *NumericalValue* or *QualitativeValue* data property. The rules that dictate the option selection are provided by *hasConditionSet* property, linking the individuals from this task to *ConditionSet*. The last one gathers all the necessary conditions through *hasCondition*. In *Condition*, it is possible to define the clinical parameter whose value will be compared, the unit it should be in and the operator that should be used. The *hasComparisonOperator* property connects individuals from *Condition* to *ComparisonOperator*. Therefore, the following individuals were created: *equal to*, *greater than*, *greater or equal than*, *less than*, *less or equal than* and *different from*.

Following a medical pronouncement, it is necessary to select the next task in the clinical workflow. Therefore there must be some kind of reference in the tasks that are up for selection (connected by the *alternativeTasks* property) to the possible results of the *Decision*. This is done through the *TriggerCondition* class that also uses the *ConditionSet.* The execution of an activity is triggered when the conditions match the decision output, the selected option. There are other classes in *ClinicalConstraintElement* that also use *ConditionSet* in a similar way, namely *PreCondition* and *Outcome*. *PreCondition* is used for all types of tasks to express the requirements of the patient state that must be met before the execution of a task. For instance, when administering some pharmacological agent it should be known that the patient is not allergic to it. The *Outcome* class puts a restriction to *Plans* and *Actions* that are oriented by therapy goals, like the case of Fig. 2 in which the *Plans* will only be considered completed when the desired levels of SBP and DBP are achieved.

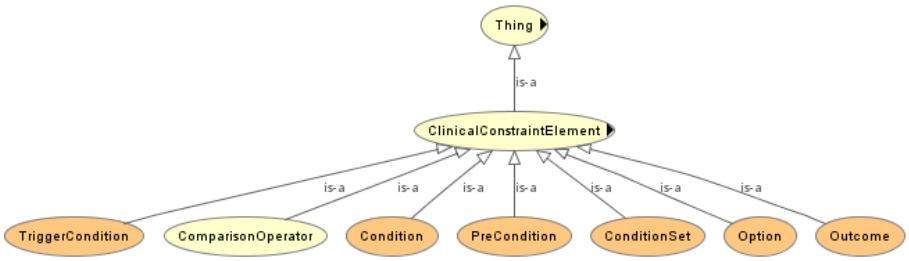

Fig. 3. Detailed view of the class *ClinicalConstraintElement* and its subclasses.

4 Discussion and Conclusions

The work presented in this paper about the CompGuide ontology reflects a different take on the representation of CPGs in machine-readable support. Although it draws some inspiration from pre-existing models [8], such as Arden Syntax, PROforma, GLIF3, Asbru or SAGE, it also introduces different views about the definition of clinical constraints, temporal properties, clinical task scheduling and how all these aspects connect with each other. Taking as a reference the oldest, and probably, the most widely (academically) used model, Arden Syntax (now a standard of Health Level 7), which represents knowledge for only one clinical decision, CompGuide provides more expressive power by allowing the definition of a clinical workflow, similarly to GLIF3 and PROforma. However, these models do not have native methods for expressing temporal constraints, using a subset of Asbru temporal language to deal with this issue. Asbru, is, by far, the model that has more temporal constructors and the most complete in this regard, but, at same time, is considered very complex and, in some cases, impractical. The temporal constructs presented in this work are intended to be a compromise between expressivity and complexity that better suits the necessities of clinical decision support systems and thus of healthcare professionals. Another important aspect is the possibility of reusing knowledge from other ontologies in CompGuide by merging the two. This way, the scalability of knowledge is

assured and the addition of supplementary information, needed for the correct application of a CPG, is enabled. None of the current formalisms for CPGs is used in a large scale for real context clinical decision support systems. Yet, there is evidence that CIG based decision support could, in fact, improve the quality of care and address the previously mentioned problems [12]. By using OWL to represent CPGs, one intends to benefit from the advantages of this knowledge representation formalism and if possible, increase the penetration of CIGs into routine medical care.

Acknowledgements

This work is funded by national funds through the FCT – Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within project PEst-OE/EEI/UI0752/2011".

References

1. Kalra, J.: Medical errors: an introduction to concepts. Clinical biochemistry. 37, 1043–51 (2004).
2. Chawla, A., Gunderman, R.B.: Defensive medicine: prevalence, implications, and recommendations. Academic radiology. 15, 948–9 (2008).
3. Hermer, L.D., Brody, H.: Defensive medicine, cost containment, and reform. Journal of general internal medicine. 25, 470–3 (2010).
4. Landrigan, C.P., Parry, G.J., Bones, C.B., Hackbarth, A.D., Goldmann, D. a, Sharek, P.J.: Temporal trends in rates of patient harm resulting from medical care. The New England journal of medicine. 363, 2124–34 (2010).
5. Field, M.J., Kathleen N. Lohr Editors; Committee on Clinical Practice Guidelines, I. of M.: Guidelines for Clinical Practice:From Development to Use. The National Academies Press (1992).
6. Vachhrajani, S., Kulkarni, A. V, Kestle, J.R.W.: Clinical practice guidelines. Journal of neurosurgery. Pediatrics. 3, 249–56 (2009).
7. Codish, S., Shiffman, R.N.: A Model of Ambiguity and Vagueness in Clinical Practice Guideline Recommendations. AMIA Annual Symposium proceedings. 146–150 (2005).
8. Isern, D., Moreno, A.: Computer-based execution of clinical guidelines: a review. International journal of medical informatics. 77, 787–808 (2008).
9. W3C: OWL 2 Web Ontology Language Document Overview. W3C (2009).
10. Neves, J.: A logic interpreter to handle time and negation in logic data bases. Proceedings of the 1984 annual conference of the ACM on The fifth generation challenge. pp. 50–54 (1984).
11. Lucas, P.: Quality checking of medical guidelines through logical abduction. Proc. of AI-2003. pp. 309–321. Springer (2003).
12. Novais, P., Salazar, M., Ribeiro, J., Analide, C., Neves, J.: Decision Making and Quality-of-Information. Soft Computing Models in Industrial and Environmental Applications, 5th International Workshop (SOCO 2010). pp. 187–195 (2010).