Ontology-driven execution of clinical guidelines

David Isern*, David Sánchez, Antonio Moreno

Universitat Rovira i Virgili, Departament d’Enginyeria Informàtica i Matemàtiques, Intelligent Technologies for Advanced Knowledge Acquisition (ITAKA) Research Group, Tarragona, Catalonia, Spain

**Abstract**

Clinical guidelines (CG) contain general descriptions, defined by health care organisations, of the way in which a particular pathology should be treated. Their adoption in daily care offers several benefits to both patients and practitioners, such as the standardisation of the delivered care and the reduction of errors, but, at the same time, there are several issues that limit their application. CGs are designed to cover a disease taking into account the available evidence but are not designed to be deployed in a particular hospital or healthcare institution. CGs include general recommendations that should be translated according to the particular settings before adoption in daily care. This adoption should also specify accountable information about the responsible actors of performing actions in healthcare teams in order to avoid errors arising during delegation/assignment of tasks. In addition, this enactment is not performed taking into account a central knowledge base or a single actor. This paper proposes the combination of a multi-agent system modelling complex healthcare organisations and knowledge representation techniques in order to build a general framework for enabling the enactment of CGs in the context of a medical centre. As a main contribution, the ontological paradigm and the expressiveness of modern ontology languages are used to design, implement and exploit a medico-organisational ontology aimed to provide the semantics required to support the execution of clinical guidelines. The knowledge-driven guideline enactment is managed by a multi-agent system modelling in a distributed fashion the clinical entities involved in the care delivery.

© 2011 Elsevier Ireland Ltd. All rights reserved.

1. Introduction

Clinical guidelines (in the following, CG) are systematically developed documents aimed to assist practitioners’ decision making about appropriate healthcare for specific clinical circumstances [1,2]. They provide very detailed information concerning the resources needed in the treatment of a patient, the decisions to be made, and the required patient’s data [3]. The adoption of CGs in daily use improves the quality of care given to the patient, minimise the variance in patient care, and cut down costs [4–6]. Several steps must be considered in the management of clinical guidelines: their authoring, their representation, their tailoring, and their execution. The first three items concern the creation and representation of CGs. Last item allows the enactment of CGs by collecting all required data and by evaluating the current status of the patient.

Although there are different attempts to create a standard language to represent CGs, none of them was successfully adopted by researchers [7]. Nowadays, there are several formalisms to encode CGs such as PROforma, Asbru, SDA* and GLARE [8–10]. In most cases, the authors of a language develop an execution engine to simulate the CG behaviour, but, as the CG represents a generic care workflow, it cannot be directly

* Corresponding author at: Av. Països Catalans, 26 (Campus Sescelades), 43007 Tarragona, Catalonia, Spain. Tel.: +34 977256563; fax: +34 977559710.

E-mail addresses: david.isern@urv.cat (D. Isern), david.sanchez@urv.cat (D. Sánchez), antonio.moreno@urv.cat (A. Moreno).

0169-2607/$ – see front matter © 2011 Elsevier Ireland Ltd. All rights reserved.

doi:10.1016/j.cmpb.2011.06.006
applied into the organisational structure of a medical centre. In consequence, medical experts should spend time tailoring CGs according to the particular circumstances of a medical centre. Such shortcomings constitute one of the barriers toward successful guideline implementations [11]. Moreover, the delivery of a particular treatment typically involves the interaction of a heterogeneous set of professionals (i.e., an evaluation unit) and distributed resources, complicating the staff management, resource allocation and the coordination of care delivery.

This paper is focused in the ontological management of CGs, which is included in a multi-agent platform (called HeCaSe2 [12]) that models the main entities involved in care delivery and permits the execution of careflow processes. Briefly speaking, the system permits doctors collecting and managing information about patients and coordinating some complex tasks, such as scheduling meetings and looking for a required service. In addition, it also aims to support the enactment of clinical guidelines by doctors in the diagnosis or treatment of diseases, by transparently coordinating all the entities involved in the process of care delivery. As it will be explained later, CGs guide the execution of care processes through different partners represented in HeCaSe2.

From the execution viewpoint, most of the CG languages store healthcare workflow information. Usually, this procedural information is described using a flowchart-like structure defining basic medical actions connected with transition dependencies between them from an initial state of the patient. However, a guideline-based execution system requires more information to understand all elements in order to decide how a CG should be executed in a particular healthcare institution. As it is analysed in [13,14], this knowledge is required during the delegation and assignment of tasks to the appropriate actors in order to avoid some medical errors (e.g., deviation from the expected course of events). This information is typically hidden, implicit and varies from one organisation to another.

In order to make this information explicit and exploitable, as a main contribution, this paper proposes to complement the information provided by CGs with an ontology, which offers a common understandable semantic framework for representing the organisational dependencies between the elements involved in the execution of CGs and their responsibilities with regards to care delivery. The proposed ontology models all the concepts and entities involved in the execution of a CG (medical terms, actions, professionals, resources, services, etc.) according to their inter-relationships and attributes. Thanks to the organisational and domain knowledge explicitly modelled in the ontology, HeCaSe2 agents can understand what and how they must perform at any moment and to which partners should request or coordinate activities.

The representation of organisational workflows and the internal relationships between healthcare entities in the ontology permits the separation of knowledge formalisation corresponding to a concrete organisation from the execution logic of CG enactment implemented by the multi-agent system. The ontology can be easily adapted to the characteristics of each particular healthcare context, without changing the behaviours implemented by software agents in HeCaSe2.

This knowledge-driven design provides some added values against related works such as flexibility, adaptability and reusability, with respect to centralised, monolithic and rigid healthcare information and decision support systems. Therefore, any organisation can have an extended and/or tailored view of the backbone ontology adapted to its particular circumstances, allowing a seamless application of supported CGs in its particular context. Moreover, as the agent execution behaviour is decided at runtime upon the knowledge provided by the ontology, the system can be immediately adapted to daily changes in the corresponding organisation by simply modifying the ontology.

The design of the medico-organisational ontology, which is detailed in this paper, follows an ontology engineering methodology aimed to identify the knowledge requirements that the ontology should fulfil in order to properly enact CGs. The compilation of those requirements required analysing the main common elements supported by CGs representation languages and the basic organisational models observed in healthcare centres (in this case, modelled by means of software agents). Finally, the ontology was used to execute in the same medical organisation several CGs defined in different languages (PROforma and SDA) to show the feasibility of the approach.

The rest of the paper is organised as follows. Section 2 makes a survey of previous works applying ontologies to healthcare systems and to supporting the enactment of clinical guidelines. Section 3 presents and formalises the ontological paradigm exploited during the ontology design and the knowledge representation possibilities of modern ontological languages. Section 4 studies the common elements included in CGs and how they are used during a treatment. Section 5 describes the architecture and the main features of the HeCaSe2 agent-based system, which models the backbone of a medical organisation. As a result of the analysis of Sections 4 and 5, the knowledge requirements collected from the care (CGs) and organisational (agents) sides are summarised. Section 6 details how the ontology has been designed from those requirements, describing the ontology engineering process based on a standard methodology and the ontology formalisation using a concrete ontological language. Section 7 illustrates with a case of study the benefits and possibilities of the proposed ontology-driven CG enactment. Section 8 details how the ontology and the approach have been validated both from the formal and practical perspectives. The paper finishes giving some conclusions.

2. Background

It has been claimed that the use of ontologies in medicine supposes an important advantage [15,16]. In this section, related works employing ontologies to support clinical guidelines will be analysed.

2.1. Ontologies in medicine

The use of ontologies to build knowledge-driven decision support systems has been widely adopted. In [17], an ontology called medical error aims to improve patient safety and to
reduce medical errors. The ontology allows healthcare professionals reporting, in a structured way, medical errors. These data are used for detecting patterns of (erroneous) behaviours, and discovering underlying factors, in order to propose solutions. In [18], authors create a taxonomy of health IT terms. In this case, the taxonomy has been included in a web site portal in order to facilitate the citizens finding information. Another implementation of ontologies in the representation of knowledge management is presented in [16]. In this case, the authors design an ontology that encapsulates all concepts and relationships involved in the treatment of hypertension. The ontology is populated with patient's data and permits the practitioner knowing exactly the current status of the patient as well as the next step to follow during the treatment. These systems exploit the ontological paradigm to build terminological structures that homogenise medical terms.

2.2. Sharing medical knowledge

Ontologies have been also exploited as mediators between execution processes and the information provided by structured or semi-structured data. In [19], the authors propose the inclusion of ontologies as an intermediate layer between electronic health records and the retrieval of the clinical data. The approach defines the ontological framework for working with the CEN EN13606 EHR recommendations. In [20], ontologies are used to share all data in a common layer. The authors propose a careflow management system and a guideline management system, and the ontology is used to represent all medical terms. Following the same idea, the K4Care project uses ontologies to exchange knowledge for delivering a set of home care services [21]. In [22], ontologies are used to represent an organisation of individual elements. With this approach, the ontology is a middleware between patients and practitioners, which is used to create virtual organisations with a common role (delivering a particular care to the patient). In this case, the ontology represents a taxonomy of concepts written in XML.

Data sharing is the first step to decentralise the execution of clinical guidelines. In this way, it allows different partners acting autonomously and exchanging data when required to follow a common goal [23]. In [24] a careflow is designed as a collection of services implemented at different sites. This is called choreography-based paradigm and creates abstractions away from entities to services allowing re-usability of processes and portability of cares.

2.3. Adoption of ontologies in the management of CGs

Focusing the attention on the management (storing and collecting data) of CGs, several authors proposed the use of ontologies to annotate parts of CGs and also to represent them entirely.

As part of a large project as DeGeL, Vaidurya is a search and retrieval engine that uses ontologies to perform a context-based search, which relies in the structure of CGs and their included medical terms [25]. In this case, ontologies are used to define a context included in guidelines stored in a repository. Ontologies include medical terms as well as relationships between them.

More related with our research, in [26], the authors propose the implementation of a task ontology called context-task ontology (CTO) that maps the knowledge required in the implementation of CGs. The ontology, represented using DAML + OIL, intends to create a CG through an independent structure that stores all concepts and relationships in a unique way. They noted that this approach had some drawbacks, such as the difficulty to defining and knowing exactly which relations are required, as well as the requirement of expert’s intervention. The same authors later described the use of ontologies to define CGs by adding a hierarchy of classes to represent medical procedures and plan [15]. However, this implies a high level of complexity as compared to flow-chart-based representations. These works suggest the use of UMLS as a central corpus of terms [27]. The work described in [28] also proposes an ontology-based representation of CGs. They propose the translation of data contained in CGs into a set of rules, which are handled by a rule-based execution engine. Again, rule definition is complex and hardly scalable in front of changes. Following the same idea, [29,30] propose to represent, using OWL ontology language, all concepts and relationships contained in a clinical guideline. The inference of the ontology permits to recommend the next action to do over a patient taking into account the current circumstances.

In [31] the authors present an interesting work that shows the flexibility obtained using ontologies in medicine. In this case, the CG is created using a graphical tool. The CG and the ontology (taxonomy) are used to create a Petri net representation, which is the input of a workflow commercial software that will enact the CG. The translation takes into account information about resources, departments, services, etc. This approach separates the information contained in a CG and how this CG is enacted in a particular location. However, with this approach, if someone changes the ontology, a translation and validation of all previously created CGs is required.

2.4. Discussion

Concerning the knowledge representation, many works show that ontologies are suitable for capturing medical knowledge in a formal way, allowing sharing and reusing it when necessary. Several works [16–18] use ontologies only to represent taxonomies of medical concepts, composing a machine interpretable thesaurus of terms. However, ontologies can be used as knowledge structures much more expressive than structured thesaurus.

With regards to the management of CGs, several researchers [26,28,30] tried to represent entire CGs using an ontological structure. This supposes the definition of ad hoc knowledge structures which, in some way, mimic the CG structure by means of a standard ontological language. Even though the use of ontologies open the door to a higher degree of inference (e.g., by instantiating ontological concepts according to concrete patient cases), the problem of CG enactment in a concrete organisation is not considered. So, the gap between the generic medical knowledge contained in CGs and their enactment in a real-scenario is still missing. Moreover, the creation of detailed ontological representations of individual CGs supposes a great amount of work both for knowledge engineers
and medical experts, being hardly scalable and making difficult the maintenance in front of changes.

Finally, there are some works [15,29] that propose the definition of organisational ontologies representing hierarchies, dependencies and responsibilities of actors involved in healthcare. Those approaches may guide the execution of healthcare processes but they rarely exploit the expressiveness of modern ontological languages in order to define complex organisational interrelations. They neither consider the relationships between actors involved in healthcare and the processes and medical entities contained in CGs in an explicit way. So, the particularities of each CG cannot be taken into consideration during their enactment.

Our approach tackles the problem in a different way. Instead of completely transforming CGs into ontological representation (which is a hard, time consuming and a hardly scalable process), it deals with CGs in their original form, maintaining the benefits of CG languages for representing medical knowledge. On the other hand, it exploits the ontological paradigm and the expressiveness of modern ontological languages to offer a detailed and structured representation of the organisational interrelations between the entities involved in care delivery in a concrete organisation. The relationships between these entities and the main concepts and processes considered in the CGs are also modelled (as responsibilities) in order to offer a tailored view of the CG enactment adapted to the organisation. The backbone of the ontology has been designed to be easily adapted in front of changes and to be extended by incorporating new CGs and medical entities. As a result, the ontology represents a background which can be exploited in a generic manner by the execution layer in order to enact CGs in a uniform way. As relationships between healthcare actors and CGs processes and entities are usually very dynamic, this knowledge-driven approach allows a seamless adaptation of the system behaviour in front of changes, bringing the flexibility and adaptability required by real-world e-health system.

3. Ontologies and representation languages

As one of our goals is to exploit the expressiveness ontologies, in this section, the ontological paradigm in which our design is based is formalised. Afterwards, the knowledge representation possibilities of the ontological language used are analysed. Those possibilities will be exploited during the ontology design (Section 6).

In [32] an ontology (O) has been defined as:

\[ O = (C, \subseteq C, R, \sigma_R, \subseteq_R, A, \sigma_A, T) \]

where

- \( C, R, A \) and \( T \) represent disjoint sets of concepts, relations, attributes and data types. Concepts (or classes) are sets of real world entities with common features (such as different types of diseases, treatments, actors, etc.). Relations are binary associations between concepts. There exist inter-concept relations, which are common to any domain (such as hyponymy, meronymy, etc.) and domain-dependant associations (e.g., an actor performs an action). Attributes represent quantitative and qualitative features of particular concepts (e.g., the medical code of a Disease), which take values in a given scale defined by the data type (e.g., string, integer, etc.).
- \( \subseteq C \) represents a concept hierarchy or taxonomy for the set \( C \). In this taxonomy, a concept \( c_1 \) is a subclass, specialisation or subsumed concept of another concept \( c_2 \) if and only if every instance of \( c_1 \) is also and instance of \( c_2 \) (which represent its superclass, generalization or subsumer). Concepts are linked by means of transitive is-a relationships (e.g., if respiratory disease is-a disorder and bronchitis is-a respiratory disease, then it can be inferred that bronchitis is-a disorder). Multiple inheritances (i.e., the fact that a concept may have several hierarchical subsumers) are also supported (for example, Leukaemia may be both a subclass of Cancer and Blood disorder).
- \( \subseteq R \) which represents a hierarchy of relations (e.g., has primary cause may be a specialisation of the relation has cause, which introduces a disorder).
- \( \sigma_R: R \rightarrow C^+ \) refers to the signatures of the relations, defining which concepts are involved in one specific relation of the set \( R \). It is required noting that some of the concepts in \( C^+ \) correspond to the domain (the origin of the relation) and the rest to the range (the destination of the relation). Those relationships may fulfil axioms such as functionality, symmetry, transitivity or being the inverse to another one. Relations between concepts are also called object properties.
- \( \sigma_A: A \rightarrow C \times T \) represents the signature describing an attribute of a certain concept \( C \), which takes values of a certain data type \( T \) (e.g., the number of leukocytes attribute of the concept blood analysis, which must be an integer value). Attributes are also called data type properties.

Additionally, an ontology can be populated by instantiating concepts with real world entities (e.g., St. Eligius is an instance of the concept Hospital). Those are called instances or individuals.

By default, concepts may represent overlapping sets of real entities (i.e., an individual may be an instance of several concepts, for example a concrete disease may be both a disorder and a cause of another pathology). If necessary, ontology languages permit specifying that two or more concepts are disjoint (i.e., individuals cannot be instances of more than one of those concepts).

Some standard languages have been designed to construct ontologies, such as DAML+OIL and OWL [33]. There are some differences between them according to their supported degree of expressiveness. In particular, OWL (Ontology Web Language)\(^1\) is the most modern and widely used one and allows, in its most expressive forms (OWL-DL and OWL-Full), to defining logical axioms representing restrictions at a class level. They are expressed with a logical language and contribute to defining the meaning of the concepts, by means of specifying limitations regarding the individual classes to which a given one can be related to. Concretely, for the OWL language, several restriction types can be defined:

\(^1\) Ontology Web Language W3C website: http://www.w3.org/TR/owl-guide/ (last access January 29, 2011).
• Cardinality: Defines that a concept’s individual can be related (by means of a concrete relation type) to a minimum, maximum or exact number of other concept’s instances. For example, certain types of Disease may have at minimum one Symptom.

• Universality (∀): Indicates that a concept’s individual has a local range restriction associated with it by means of a certain relation type (i.e., only a certain class of individuals can be the range of the relation). For example, all Symptoms of a certain Disease must be of the same type.

• Existence (∃): Indicates that a least one concept’s individual of a certain class must be the range of a relation for another concept’s individual. For example a Disease always presents a certain kind of Symptoms, even though other ones may also appear.

All those restrictions can be defined as Necessary (i.e., an individual should fulfil the restriction in order to be an instance of a particular class) or Necessary and Sufficient (i.e., in addition to the previous statement, an individual fulfilling the restriction is, by definition, an instance of that class). This is useful for supporting reasoning when dealing with unknown individuals.

In addition, OWL permits the representation of more complex restrictions by combining several axioms using standard logical operators (AND, OR, NOT, etc.). In this manner, it could be possible to define, for example, a set of Symptoms which co-occur for a particular Disease using the AND operator.

OWL and more concretely OWL-DL has been selected to implement the proposed ontology as it offers a high level of expressiveness based on Description Logics and, at the same time, allows performing automatic reasoning aimed to check and detect inconsistencies in the ontology (unlike the most expressive form OWL-Full, which does not ensures the decidability and the computational completeness of the language).

Moreover, OWL is endorsed by the W3C and well supported by ontology editors and knowledge tools.

4. Clinical guidelines: definition and execution

A CG is a systematically developed statement designed to assist clinicians in making decisions about health care for specific clinical circumstances [2]. Most of the CGs are published as text and typically include criteria describing their applicability to particular groups of patients, the recommended processes of care and appropriate use of materials and procedures, as well as providing ancillary information such as supporting evidence [3]. The use of computerized decision support systems can add value to textual documents by actively offering evidence-justified and patient-specific advice at the point of decision-making [34].

From an AI perspective there is an alternative to publishing guidelines solely in human readable form such as text, tables or flow diagrams, which is to formalise the medical knowledge contained in these guidelines in a machine-readable format that a computer can use to support clinicians in their routine work. Many researchers have implemented methods for interpreting and transforming parts of narrative texts using an interpretable guideline formalism. These methods are usually based on supervised annotations (mark-ups) performed by medical experts (e.g., [11]) and may include validation mechanisms to avoiding inconsistencies between logical conditions (e.g., [35]). The result of this transformation process can be expressed in different formal languages, such as Asbru, EON, GLIF, PROforma, SDA* and GLARE. These languages were analysed and compared in [9,36]. The adoption of one formalism as standard should improve the interoperability between different tools and systems, however, none of them was widely adopted by researchers. Although each language is designed with a different coverage and particularities (e.g., triggers in the case of PROforma, inclusion of roles in SAGE, or the use of synchronisation functions in GLIF3), most of them share a set of basic functions [7,9]:

1. Mechanisms for coordination or synchronisation of actions.
2. Ability to create sub-plans or sub-guidelines (so that different levels of abstraction can be considered when working with guidelines).
3. Possibility of storing the state of a guideline which is being executed (e.g., awaiting for data, suspended, finished or it cannot be accomplished taking into account the state of the patient).
4. Definition of required (clinical) data.
5. List of actions to be performed during a treatment.
6. Description of all decisions to be taken in a certain point taking into account different criteria.

From the knowledge representation viewpoint, these components can be divided into two main groups. In our approach, the first set (items 1–3) represents the procedural side of the execution and it is used by the guideline execution engine to go throughout the CG step-by-step checking both medical-related dependencies and constraints between nodes in order to recommend accessible nodes. In consequence, this information can be collected directly from the CG. The second set (items 4–6) includes medical terms involved in healthcare as well as how they should be handled in a real setting. This set is related to how a CG is enacted in a particular healthcare organisation.

All these elements are shown in the CG depicted in Fig. 1. In this case, the CG for the prevention of coronary heart disease (CHD [37]) is written using PROforma syntax [38], which is one of the best well-known languages for representing CGs. Most of the arrows shown refer to synchronisation (item 1) points between different tasks (for instance, taking into account the value of a decision, choose an option or another). Moreover, coordination issues occur when allocating tasks to handle over a patient. Referring the 2nd item, the presented CG shows two different granularities of elements: on the one hand, a first level of elements show the general treatment of the CHD (depicted as rectangles in Fig. 1), and on the other hand, some complex and/or specialised tasks as referral a patient to a specialist are hidden in a particular sub-plan (depicted as rounded rectangles in Fig. 1). Concerning the state of a task (item 3), PROforma defines four main stages (dormant, in progress, completed or discarded), which are monitored by the execution engine over the treatment. One of the advantages of the adop-
The example describes basic decisions (depicted as circles in Fig. 1) such as the evaluation of findings, and complex decisions such as the evaluation of the patient life factors across a standardised scale.

5. **HeCaSe2: a multi-agent system for healthcare service delivery**

HeCaSe2 defines an architecture of software agents implementing different roles and supporting multiple interactions
between them and humans, medical devices, and an electronic medical record [12]. They are structured based on the organisational rules of a medical centre (in this case, following a hierarchical-based topology [39]). Agents are autonomous software components representing real world entities (e.g., doctors, departments, and devices in a medical centre) behaving according to their own knowledge and data. The system is decentralised and the agent configuration (number and type) depends on the specific structure of a concrete medical organisation. As this configuration is the base of the CG enactment in a particular scenario, the organisational dependencies between all these entities should be included in the ontology in order to, for example, associate medical actions with those responsible for carrying them out.

The basic architecture of the multi-agent system (MAS) is shown in Fig. 2. The user (a patient) is located at the top of the architecture that interacts with the system through his User Agent (UA). This agent stores static data related to the user (e.g., national health care number, name, address, access information—login, password, and keys) and dynamic data (the timetable and the preferences of the user). The Broker Agent (BA) is an agent that knows all the medical centres located in a certain area.

A Medical Centre Agent (MCA) centralises and monitors the outsider accesses to the agents that manage the information of a medical centre. A MCA includes different type of agents: some Department Agents (DAs) and some general Service Agents (SAs). Each DA represents a department of a healthcare organisation and has a staff of several doctors, modelled through Doctor Agents (DRAs). As it can be shown in the general architecture of HeCaSe2, DA is a gateway between the elements included in a department and the rest of the system, and permits to filter some incoming requests that are not addressed to the agents contained in it. A SA (located at any level) represents a physical resource embedding its own characteristics and constraints. Two types of services are distinguished: generic and particular services, which can be used by all partners of a medical centre (e.g., blood test service) or localised in a particular department (e.g., thermograph), respectively. Both MCAs and DAs are aware of the services they can provide (when a SA enters the system, it sends a message detailing its services to the associated MCA or DA).

Each department contains a Guideline Agent (GA) that performs all actions related to guidelines (e.g., it can retrieve the CG associated to a specific illness). It relies on specific CG language wrappers in order to parse CGs and extract actions, decisions, etc. The GA contains CGs associated to the department where it is located. In addition, each department includes an Ontology Agent (OA) which provides access to the knowledge stored in the ontology in order to properly enact CGs. A DRA proactively requests the meaning of terms during the enactment of CGs as they are modelled in the ontology.

This distributed approach is flexible and dynamic, and permits all partners interact between them in order to find the information. When a doctor identifies a required service for a patient, the corresponding DRA asks its DA in order to identify if the required service is available in the same department, if not, this message is sent to the rest of departments and also to the MCA. In this negotiation the preferences of the user are taken into consideration in order to collect the most preferable proposals. Each SA stores an agenda of all arranged scheduling and it sends a list with free slots taking into account (as more as possible) the received constraints. At the end of this negotiation, the DRA sends the acceptance of one proposal and rejects the rest. The selection of the best alternative is made by the human patient when asked by the practitioner.

At the bottom of the architecture, a Medical Record Agent (MRA) controls the access to a database that stores all medical records of the patients of the medical centre. The database stores all results and findings inserted by practitioners and service agents related to each patient. Appropriate security measures have been adopted to ensure that only properly authenticated and authorised agents may access and update the medical records. The MRA wraps the flow with the databases where the data are located (in one node or scattered among local medical centres).

The coordination of the tasks to be performed is made by the DRA under the supervision of the doctor by retrieving all information concerned to the patient and suggesting the execution of the steps contained in the CG applied to him. It is important to note that the enactment of a CG is performed in a supervised manner, suggesting and assisting the doctor about the steps to be performed according to what the CG states. Some actions such as resource discovery or allocation, actor coordination or data access are automated but the final decision or action is taken or performed explicitly by the doctor.

During the visit, the doctor can refer the patient or to ask for the execution of a procedure suggested by the CG enactment engine. Some languages distinguish three main elements into a CG: decisions, medical actions, and enquiries. HeCaSe2 also adopts the notion of sub-plan described in PROforma and SAGE, which can be used optionally by medical experts.

In the case of decisions, HeCaSe2 shows the human expert the logical conditions included in the decision and presents alternatives to follow. This allows the practitioner to take and informed decision. After the acknowledgement of the practitioner, the CG enactment engine continues with the next step to follow.

In case of medical actions, this referral comprises two steps: to know who the responsible actor is, and then to negotiate the action execution. First of all, the DRA contacts with the OA in order to know the kind of agent able to perform the task (a professional or a service), and then the DRA negotiates the best alternative according to the available agents of that type, the preferences/constraints of the user, the doctors and the medical centre services. A user profile containing the set of preferences about medical centres, doctors, and other criteria used to guide the scheduling are stored in his/her corresponding UA. When the action is completed, the results are sent automatically from the SA that has the executed task to the MRA that stores them in the patient’s medical record. The interesting point is that all the scheduling processes can be made automatically by the autonomous agents. This avoids the patient having to waste time and effort to make a particular booking for each needed test or examination, or the doctor having to worry about asking to a service for the results of a test.

Enquiries describe a set of variables required during the CG execution (e.g., lifestyle factors, a level of blood pressure,
6. An ontology to support the execution of CGs

In the following, the design of the ontology is explained. It aims to provide the semantics needed to fill the gap between the medical knowledge represented in CGs (introduced in Section 4) and their agent-based execution (introduced in Section 5) in a concrete organisation. First, the iterative ontology engineering process, based on a standard methodology will be described. Then, the ontological components and their interrelations expressed in terms of the ontological language will be detailed.

6.1. Ontology engineering

The construction of an ontology modelling a real scenario is a complex task, as the completeness of the ontology should be ensured even when implicit knowledge makes difficult the formalization process. Moreover, there does not exist a unique way for modelling ontological knowledge. In this sense, Ontological Engineering provides methodologies, such as METHONTOLOGY or ON-TO-KNOWLEDGE [40], for building ontologies in a comprehensive and structured way. From all of them, the 101 method [41] has been selected as it describes an iterative approach which allows designing and refining the ontology in several incremental steps. Considering that the ontology has been designed from the analysis of a significant amount of distinct CGs, an incremental engineering approach fits very well with the need of extending the modelled knowledge as new CGs are incorporated. Moreover, it proposes a flexible workflow in which ontological engineering steps can be re-executed in no particular order and it is independent findings, etc.). These values can be provided by the practitioner that enacts the CG (most of times) or taken directly from the patient’s health record, but in other cases they may require their referral (when the value is provided by a specialist). In last case, the DRA contacts with the OA in order to know the kind of agent able to provide the value, and then, the DRA negotiates an appointment with one of those agents.

Finally, the CG contains sub-plans that embed more complex tasks or procedures. A plan contains a set of the tasks in the same way than the ancestor plan. As an example, the process of ordering laboratory tests that require the description of the target parameters to analyse and are hidden the details about how they are conducted. Another example is referral of a patient to a cardiologist that require details about the current treatment and the purpose of the appointment like the evaluation of the patient or follow-up. Basic elements contained in a sub-plan are conducted in the same way than the rest of elements of the guideline.
from the ontological language. In the following, the aims of each engineering step and the decisions taken and results obtained for our concrete case are detailed.

The 101 methodology divides the ontology construction process in several iterative steps:

1. **Determine the scope:** The goal is to bind the ontology domain according to the knowledge requirements. It is proposed to sketch a list of questions (competency questions) that a knowledge-based system will be able to answer by consulting the ontology.

   In our case, the competency questions resulted from the knowledge requirements compiled during the analysis of the CGs models (Section 4) and the organisational structure of a medical centre (modelled by means of agents as described in Section 5). After the analysis, the *medico-organisational* ontology should provide answers to the following questions:
   - Which daily care *procedures* are provided by the system?
   - Which patient’s clinical data are managed by the system?
   - Which actors participate in the provision of those care procedures? An actor will be associated with every sort of healthcare entity included in the structure of the medical centre.
   - How these actors are organised into a medical centre? There are *departments* that are composed by different kinds of *practitioners* and *specialists*, and also *services* that are composed by medical devices performing basic tasks.
   - Which actor is the responsible for acquiring a certain data or performing a certain activity, which are required during the CG enactment?
   - Which are the possible results of a medical procedure, which variables should be filled up as a result of an action and which the vocabulary and terminology related to the enactment of a guideline are? It is important to note that the modelled medical knowledge does not intend to offer a complete and detailed view of an aspect of the medical domain (i.e., a *domain ontology* such as UMLS) but to reflex the medical scope of the considered CGs in order to support their enactment.

2. **Consider reusing existing ontologies:** It is always worth considering if a previously development ontology fits with some of the knowledge requirements. In the medical field, many domain ontologies have been developed [15]. They are mainly employed to define medical terminology with a clear and non-confusing meaning, which facilitates the exchange of information between different systems. Some examples of widely used medical ontologies focused on structuring domain terminology are GALEN, MeSH (Medical Subject Headings), SNOMED (Systematized Nomenclature of Medicine, Clinical Terms), UMLS (Unified Medical Language System), and ONS [42]. However, it is worth to note that most of these domain ontologies focus only on the terminological and taxonomical aspect of the medical knowledge (i.e., defining one or several concept hierarchies of diseases, syndromes, treatments, etc.), thus, they do not model organisational workflows or entities’ interrelations.

   As our ontology requires to include some medical concepts (e.g., findings, medical actions, procedures, etc.) found in CGs in order to associate them with the actors involved in care delivery, the medical classification codes defined by the UMLS semantic network [43] are also included. It permits to associate a set of semantic types (e.g., findings, behaviours, entities, etc.) to each of medical term. In this manner, each medical concept extracted from a CG is properly classified and can be directly mapped with the knowledge modelled in a wide and detailed medical repository such as UMLS.

3. **Enumerate relevant terms:** Taking into account the competency questions formulated in the first step, their answers will generate terms from which one would generate statements.

   All medical terms used inside CGs as well as, all organisational terms involved in the multi-agent system (agents, departments, and services) are listed. Properties used to interrelate or define those terms were also enumerated.

4. **Define classes and hierarchies:** Assuming the common nature of many of the terms enumerated in the previous stage, they can be organised in several hierarchies: CG related terms (such as medical entities, procedures, and personal data) and agents (professionals, patients, departments or services). A top-down development process has been performed, starting with the definition of the most general concepts and recursively incorporating class specialisations. In our case, the knowledge structure used is a taxonomy.

5. **Define properties:** Class definitions are not enough to answer the competency questions formulated in the first step. The internal structure and inter-relations of concepts should be also described.

   Concept relations and attributes listed in step 3 are modelled as ontological properties. Three types of properties have been defined. First, object properties are meant to interlink class instances allowing modelling which the responsible of a CG element is, where an agent belongs to, to which procedure a medical concept is associated or which kind of results are obtained from the execution of an action. Secondly, data-type properties linking class instances and value literals allow indicating which the result of an action is. Finally, annotation properties allow defining attributes at a class level. They are useful to associate medical concepts to standard medical codes or to state different textual labels referring to the same concept (i.e., a list of synonyms, the corresponding code unique identifier).

6. **Define restrictions:** As introduced in Section 3, OWL-DL allows modelling complex restrictions to bring the necessary expressiveness to class definitions.

   Existential and Universal restrictions have been defined at class level in order to indicate to which department an agent belongs, which the responsibilities of an entity are or which the expected results of an action are. Appropriate domains and ranges and axioms have been defined for each property.

   All mentioned steps are meant to be executed in an iterative way in any particular order, allowing creating the final ontology in a flexible and incremental way. So, after each repetition, a more refined and detailed ontology is obtained.
In our case, the analysis of additional CGs resulted in new knowledge to be modelled over the basic ontological structure (i.e., class hierarchies and main properties), resulting in new class specialisations and restrictions. The backbone structure of the ontology (i.e., main root classes and class restrictions) is maintained intact allowing a homogenous management of the ontology by the execution layer and a seamless incorporation of changes and extensions thanks to the knowledge-driven design.

6.2. Ontology formalization

As a result of the ontology engineering process, in this section, a detailed description of the backbone elements of the ontology formalised in OWL-DL is provided.

The ontology models three main elements: types and organisational structures of agents, all possible semantic types (divided into entities and events) to which the modelled medical concepts may correspond as provided by UMLS, and different types of elements extracted from CGs which are associated to their responsible agent (see Fig. 3). Formally, the ontology has 4 root classes (Agent, Entity, Event, and Guideline element), which are hierarchically specialised according to the structure of the medical centre and the CGs incorporated into the ontology. All class restrictions are defined only as Necessary because class inference is not applied at instance level. In the following, their characteristics and interrelations defined by means of OWL properties and ontological restrictions are presented.

6.3. Agent

The Agent class represents the software entities included in the multi-agent system (as described in Section 5), which correspond to a certain type of real-word entity. It is specialised into five main subclasses:

1. **Patient**: Representing generic patients. In the multi-agent context, they correspond to User Agents.
2. **Practitioner**: Representing non-disjoint generic professional roles found in a medical centre (i.e., concrete professionals may play several roles). According to the centre’s structure, this class can be further specialised in order to incorporate concrete types of healthcare professionals (e.g., cardiologist, oncologist, nurse, etc.). The class incorporates restrictions in order to establish the roles, duties and skills of each type of professional. This class represents Doctor Agents in the multi-agent system context.
3. **Medical centre**: Representing a healthcare organisation. It corresponds to Medical Centre Agents.
4. **Department**: Representing disjoint parts of a medical centre with a specific responsibility (e.g., cardiology, surgery and intensive care unit). It is specialised according to the departments’ types found in the concrete organisation. It corresponds to Department Agents.
5. **Service**: Representing a set of general services linked to human or physical resources used during diagnosis or treatment (e.g., blood test, radiography, an electrocardiogram, etc.). It represents Service Agents.

Concrete agents running in the multi-agent system of a medical organisation would represent the instances of the classes modelled in the ontology (i.e., individual Family Doctors or concrete Cardiology Departments). In this manner, agents can consult their corresponding class in the ontology at execution time in order to be aware of their organisational interrelations.

Relations defined for Agent subclasses are inspired in usual healthcare organisations. Concretely:

- **belongsTo** (establishing Service, Practitioner and Department as Domain and Medical centre and Department as Range): States the organisational dependencies between entities. A transitive axiom states that if a Practitioner belongsTo a Department and that Department belongsTo a Medical centre, then the Practitioner belongsTo the Medical centre. Universal restrictions are defined at class level in order to indicate, at an instance level, the corresponding dependencies. For example, for X-Ray, $\forall$ (belongsto (Radiology states the department to which a service belongs.; for Nurse, $\forall$ belongsTo (Department states that nurses may belong to any department; for Family doctor, $\forall$ belongsTo (General medicine \cup Emergency \cup Paediatrics) indicates that a family doctor may belong to any these three departments.
- **isComposedBy** (Medical centre and Department → Service, Practitioner and Department): States the inverse relation to the belongsTo property. An inverse axiom is defined at property level in order to maintain the coherency and to allow an easy browsing in both directions of the relation, retrieving the proper elements contained in a department or medical centre.

A basic set of departments, practitioners and services typically found in a medical centre are defined. However, it is possible to represent different variations by creating or modifying subclasses. For instance, paediatrics department in a hospital may be different from a primary attention centre located in village or covering a part of a city. Usually, the former has more instruments used for diagnostic and other therapeutic purposes than the latter.

6.4. Entity and Event

Entity and Event are disjoint hierarchies of classes concerning the different semantic types of medical entities. They were picked from UMLS Semantic Network [43]. Currently, UMLS distinguishes 135 different semantic types divided in those groups. Entity defines a broad type for grouping physical and conceptual entities, and Event describes a broad type for grouping activities, processes and states.

The UMLS semantic network [43] defines two different relationships between classes: associated-with and is-a. The former explains that two elements have a significant or salient relationship to (e.g., Finding is associated with Anatomical abnormality and Pathologic Function is associated with Educational Activity), and the latter describes that an item is more specific in meaning than his parent (e.g., a Health Care Activity is an Occupational Activity, or that an Occupational Activity is an Event). In our case, only is-a relationships are used because associated-with is intended to express a network of general axioms with
different purposes related to the knowledge embedded into CGs.

Semantic types are associated to classes modelled in the ontology in order to be able to map them in a wide and detailed medical repository such as UMLS. This brings coherency to the modelled knowledge and allows properly defining and contextualising the meaning of elements found in CGs.

The relationship between the elements modelled in the ontology and the UMLS semantic type is defined by means of the hasSemanticType property, where Domain is Agent and Guideline element and Range is Entity or Event. As the semantic type is unique, a functional axiom is defined for that property. An existential restriction is defined at a class level in order to state the concrete semantic type of a class, such as Increase physical activity is a finding associated with life-style but Physical activity is a daily or recreational activity that should be treated inside an action.

6.5. Guideline element

The last part of the ontology represents the main elements involved in the execution of concrete CGs and associates them to their responsible agents. All those elements are grouped in a general Guideline element class that is specialised into three main disjoint subclasses:

1. Medical entity: Represents any medical element found in a CG, such as diseases or symptoms, which are needed for diagnosing or defining the status of the patient or proposing a treatment.
2. Procedure: Represents all diagnostic and treatment procedures performed over patients.
3. Personal data: Represents concepts related to the patient state or attitude (e.g., sex, smoking, age, lifestyle criteria) that may be relevant for a treatment.

The following relations are defined for those classes:

- hasResponsible (Guideline element → Agent): States the agent which is responsible, in a concrete medical centre, of an element of the guideline. An Existential restriction is defined at a Guideline element class level with regards to the Agent or set of Agents which are in charge of dealing with that element (e.g., testing for a medical condition, evaluating a symptom, compiling patient's personal data, etc.). For example, for Mammography, ∃ hasResponsible Radiology states that the diagnostic procedure consisting on performing a mammography is performed by the radiology department.
- isResponsibilityOf (Agent → Guideline element): States the inverse relation to the hasResponsible property. An inverse axiom is defined to maintain the coherency, allowing knowing the responsibilities of a practitioner, department or service.
- hasResult (Guideline element → dataType): Is a data type property that indicates the expected result of the management of a Guideline element (e.g., the value resulting from a blood pressure test, the name of the patient, the result of an incisional biopsy). As resulting data may have different data types (e.g., Boolean, number, enumerate, etc.), this property is specialised in the following subproperties: hasResultBoolean, hasResultEnumerate, hasResultString, and hasResultInteger, hasResultComplex.
- hasAssociatedProcedure (Medical entity → Procedure): States, for some Medical entities which need to be evaluated by means of a medical procedure which is that concrete procedure.
An Existential restriction is defined at class level to state this relation. It is used only to audit which are the medical procedures associated to a disease and to ease the collection of data provided by DRA. In this case, when enacting a CG, this information is used to recommend other procedures included in the ontology and related to the medical concept, such as blood pressure has associated its measurement.

One of the benefits of exploiting an ontology is the use of a common comprehensive vocabulary. As CGs are incorporated into the ontology, new elements are needed to be modelled. In some cases, some concepts already exist in the ontology but with a different name (according to the terminology used for different ontologies). In order to state the different textual labels used to refer to the same concept, two annotation properties are included at a class level: hasCUI states the concrete Code Unique Identifier defined by UML5 in order to localize and define the concept in that repository; hasSynonym declares a list of equivalent labels used to refer to the same concept in different CGs. This increases the interoperability between CGs referring to equivalent concepts expressed with different notations and eases the re-use of already modelled knowledge. In fact, knowledge re-use is a key aspect of this modelling process because, as new CGs are incorporated in the same organisational context, the need for concept modelling is reduced to the new concepts, simplifying the knowledge representation process (more details in Section 8).

7. Enactment of a clinical guideline: prevention of coronary heart disease

Previous sections have introduced the main elements of our platform: HeCaSe2 as the agent-based platform that acts on behalf actors during the enactment of CG, and the medico-organisational ontology, as the element that links those elements in order to customise their execution. In order to illustrate how those elements work together, this section details the enactment of a CG for the prevention of coronary heart disease (CHD). The CG is written using PROforma syntax [38], and Tallis engine2 has been used to load and parse it. The whole guideline is shown in Fig. 1. This CG is one of the developed in the K4Care project [37] and aims to provide a series of evidence-based recommendations related to coronary heart disease and its secondary prevention (for patients already after myocardial infarction).

From a general viewpoint, the enactment of the guideline starts when the doctor, during a medical visit with the patient, selects the appropriate guideline for the diagnosed disease. The DRA shows an up to date list of available CGs received from its departmental GA. HeCaSe2 allows the doctor browsing, visualizing and selecting guidelines that are retrieved from the GA. When the DRA receives the CG selected from the doctor, the DRA initialises the CG execution module with the CG code and all patient’s data retrieved from his EHR. With this information, the execution module is able to initiate the enactment, whereas the DRA remains waiting guidance from the practitioner when needed. On the one hand, the DRA interacts with the practitioner and, on the other hand, the execution module is aware of the current state of the patient in the CG, it retrieves values about patient’s examinations, and forwards the DRA with the next action or decision to be performed or asked.

During the enactment of a CG, the execution module deals with three basic elements and other sub-plans:

1. Decisions. The execution module informs to its listener (DRA) that a decision has been reached. The agent retrieves the information related to this decision, such as the attached logical condition(s), related medical data and the result(s) of the evaluation(s). The DRA forwards them to the medical expert through his graphical interface which evaluates them in order to take a decision in a supervised and informed manner. The logical conditions embedded into the decision are composed by variables with particular values (e.g., if level of systolic blood pressure is lower than 120 and the diastolic is lower than 90). All values of those variables should be available before reaching the decision, embedded into a previous enquiry, as it will be shown in the following.

2. Actions. The module informs that an action should be performed. First, the DRA must search for an entity able to execute this action. This information is obtained from the medico-organisational ontology according to the actor types and responsibilities modelled in the ontology for the particular centre. As a result, the agent knows which kind of actor is responsible of performing the action. Then, the DRA finds a concrete actor within the appropriate department, inside or outside the centre. As it has been explained in Section 5, this task requires the negotiation among different partners that are able to accomplish this task. When the task assignment (e.g., an appointment between the patient and a SA) is performed and the action (e.g., blood test) is executed, the SA sends the results to the MRA that stores them in the patient’s EHR. In the next meeting between the patient and the practitioner, the execution engine and the DRA will be able to retrieve these values if required. When actions should be performed by the same practitioner that enacts the CG (e.g., add aspirin), the execution module directly forwards the action to the DRA and waits for the acknowledgement of the practitioner before continuing.

3. Enquiries. An enquiry contains a requested set of variables (e.g., a list of life style factors, the current level of blood pressure, etc.) that may be needed in a further step (e.g., a decision). These data are searched in the EHR of the patient via the MRA. If a value of a variable is not found, it should be collected in the same way as actions: the medico-organisational ontology describes the concept corresponding to that value and designates who is the responsible. On the one hand, if the value cannot be provided by the initiator of a CG, HeCaSe2 performs an agent-based negotiation between some agents able to provide the value and finally selects one. After the appointment, the required value is provided by the entity (i.e., medical service or medical staff) through its correspond-
ing graphical interface, and stored in the EHR of the patient. On the other hand, if the enquiry contains elements that can be satisfied by the same practitioner or those elements are already available in the EHR, the same practitioner can insert or validate the values. As soon as all values embedded into an enquiry are available, the execution can continue.

4. Plans. A plan embeds a therapy to be conducted over time. A plan is a container of atomic elements (decisions, actions and enquiries) or other complex plans. It permits to describe different levels of abstraction in the root plan and hidden administrative or sequential procedures that can be enclosed in sub-plans. From the execution perspective, the execution engine starts the execution of a sub-plan in the same way than other plans, locating the first element to handle and acting accordingly.

With regards to the CG introduced at the beginning of the section, CHD is the end result of the accumulation of atheromatous plaques within the walls of the arteries that supply the myocardium (the muscle of the heart) with oxygen and nutrients. Most individuals with a coronary heart disease show no evidence of disease for decades as the disease progresses before the first onset of symptoms, often a “sudden” heart attack, finally arise. The disease is the most common cause of sudden death, and is also the most common reason for death of men and women over 20 years of age and it is a chronic illness that requires continuing medical care and patient self-management education to prevent acute complications. The therapeutic options for the treatment and prevention of this illness are based on four principles: life style changes, medical treatment (drugs), coronary interventions as angioplasty and stent-implantation, and coronary artery bypass grafting. Acute deteriorations in patients with coronary heart disease should be managed in a hospital setting unless the patient has already commenced on a palliative care pathway. In the CG, the prevention of the coronary heart disease is divided into two main parts. The former includes the less severe actions aimed to discard other related pathologies such as diabetes. The latter, focused on patients diagnosed with diabetes in the first stage, includes the prescription of particular drug therapies.

Usually, the CHD is a prevention guideline started by a family doctor (which will be its owner). As stated above, the first step consists on collecting all available data from the patient’s medical record through the MRA. In this example, it is assumed that the patient starts the treatment without past results available in his/her medical record.

If we observe the guideline (Fig. 1), the first node is an enquiry that includes a list of parameters related with the patient’s life style. At this point, the DRA contacts with the OA in order to know all details about these concepts from the ontology. The first term is “Weight reduction”. The OA identifies this term as a synonym of the UMLS concept named “Body weight decreased” with the UMLS code “C0043096”. Following the relationship hasSemanticType, the agent knows that it is a “Finding”. In addition, the agent follows the property hasResponsible and it retrieves that a “Family Doctor” or a “Nurse” are able to perform it as well as this item hasBoolean as expected result. As said previously, the owner of the CG is a family doctor which started the process and, in this case, he is able to insert the value required by the enquiry after examining the patient. The DRA forwards the question to the practitioner that inserts the value. The rest of items embedded into this enquiry are handled by the same practitioner.

When all the values are inserted, the execution module jumps to the next CG node: a decision point. In the case of PROforma, a decision has slots defining the candidates that the decision should choose between, the logical conditions which should be evaluated when formulating arguments for and against the options, and the criteria for choosing specific candidates [38]. The execution module enacts the decision and all the information is shown to the human expert through his graphical interface highlighting the satisfied candidates according to available data. For instance, if the patient has bad habits and attitudes, the practitioner should stop the CHD and change the life style of the patient. If we assume that life style factors are correctly evaluated by the practitioner, the next step of the CG evaluates other risk factors of this disease: a high blood pressure in combination with diabetes. The measurement of blood pressure is an action that results in a two values as diastolic and systolic blood pressure values collected in a string. The concept “blood pressure” has the associated procedure (hasAssociatedProcedure) concept “Blood pressure measurement”, which is achieved by a “family doctor” or a “nurse”. The result of this element is a string (hasResultString) with the form diastolic/systolic. The enquiry is completed with the concept “diabetes mellitus”. This last concept has a Boolean value (hasBooleanResult) that is responsibility (hasResponsible) of a “family doctor”, who should diagnose it. In this manner, the ontology indicates the DRA asking for all these values to the same family doctor that enacts the CG. As shown in the previous enquiry, these values are inserted and validated by the practitioner, stored into the patient’s health record, and available for further evaluations.

A goal of the CHD guideline is to stabilise the patient with a level of blood pressure of 140/90 or less in the case of diabetic patients. Medication is usually needed when life style changes are ineffective in keeping the blood pressure within normal range. So, the next decision in the CG evaluates if the patient has a high blood pressure. If the patient has an abnormal blood pressure, the CG begins a medication sequence starting with the action named “Start with ACE inhibitors”. The information included in this action is interpreted by the OA in order to know how to proceed. The ontology stores that ACE is the acronym of the concept “Angiotensin-Converting Enzyme Inhibitors”, with the UMLS code “C0003015”. This concept is a “Pharmacologic Substance” and is responsibility of a “family doctor” or a specialist (“cardiologist”). In consequence, the prescription of this drug is requested to be performed by the same family doctor that enacts the CG and the execution module waits the confirmation of its completion before continuing. The practitioner that enacts the CG should determine the period of time before re-evaluating the patient in order to know the effectiveness of the prescription.

As shown in the CHD CG, when the blood pressure is stabilised (with or without medication), the patient is referred to a specialist to evaluate his/her current status (sub-plan named “Appointment specialist (cardiologist, 3-6 months”)). The appointment requires a deep explanation of the reasons
to refer the patient to a specialist. In this case, the patient requires a new evaluation in order to confirm or not the first diagnose. The plan includes a first enquiry with the parameters of the appointment and the action “referral specialist”. In this case, the execution module interprets the action in order to extract the concept “cardiologist” and the period of time “3–6 months”. As stated in the ontology, the cardiologist is a “practitioner” which belongs to a cardiology department (Cardiologist belongsTo Cardiology, which is a department). At this point, the DRA should contact with a cardiologist (it looks for available cardiologist agents, and then, it begins a negotiation with them in order to pick a concrete one) and books a meeting for the patient. HeCaSe2 performs an agent-based negotiation taking into account temporal constraints (if they are required) such as this case. The period of time “3–6 months” is used to constraint the proposals arranged between two DRAs (on behalf of the practitioner and the cardiologist, respectively). The agreement is made after receiving the acknowledgement of the human specialist, and then, the enactment is halted until the visit is performed. During the medical visit, the specialist will introduce in the EHR his evaluation of the patient. After that visit, the family doctor will have the results of the evaluation of the specialist and the careflow may continue.

As a result of the evaluation, if the patient has a normal blood pressure without diabetes, the practitioner should evaluate other parameters obtained from a general purpose blood test. This procedure is a sub-plan defined to obtain a wide range of values required later. In this case, the sub-plan includes an action named “blood test”. The OA retrieves from the ontology that a “blood test” is a “laboratory procedure” which is a responsibility of a “laboratory analysis”, which is a subclass of “service” agent. The laboratory is a shared service of a medical centre (“laboratory analysis” belongs to “medical centre”), and the DRA should look for it directly at medical centre level. The DRA asks its medical centre agent the name of the SA who is a “laboratory analysis”, in order to negotiate an appointment for the patient. After performing the blood analysis, the SA will insert the results into the EHR via the MRA. These results include values such as the level of blood glucose (HbA1C) and the level of triglycerides, which will be available to the practitioner in the next medical visit. All SAs have an internal configuration with particular descriptions of all covered tasks. In the case of a blood test, an internal configuration defines all the values resulting of this test, which may include more than those required in the CG.

The next node of the treatment is an enquiry that looks for different values (HbA1C, triglycerides, non-HDL-C, and LDL-C). As those values have been set by the result of the execution of the blood test, they are available in the EHR for the CG execution engine that forwards them to the human practitioner. These values are confirmed by the practitioner before continuing with the next node of the CG. As values are stored in patient’s EHR, the results of a concrete procedure can be analysed in different points of care during the treatment.

The rest of the CG is intended to determine the best drug therapy for the patient taking into account drug interactions (e.g., anticoagulants and aspirin).

During the whole treatment, if a result is anomalous, or the diagnostic is erroneous, the practitioner can decide stop the execution of the CG in order to refer the patient to a specialist or to change the treatment.

8. Ontology validation and maintenance

The validation of the designed ontology has been conducted from two points of view. On the one hand, the formal coherency of the modelled knowledge has been tested by means of a reasoner. As stated in Section 3, OWL-DL ontological language provides a high expressiveness and allows executing automated reasoning aimed to check the consistency of the modelled entities, relations and restrictions. Concretely, a reasoner can check whether or not all the statements and definitions in the ontology are mutually consistent (i.e., the fact that a class can be instantiated). FaCT++3 was used as a reasoner during the ontology development process.

On the other hand, the practical feasibility of the ontology has been tested by including it into the HeCaSe2 multi-agent system and executing a set of CGs. In order to configure a realistic healthcare scenario, the structure of real medical centres of a city (Reus, Catalonia) has been modelled. Concretely, one hospital (with 7 departments) and six primary attention centres (with 4 departments per centre) have been modelled by means of agents and formalised in the medico-organisational ontology. As a result, 60 agents representing practitioners deployed through these centres have been created. In addition, 15 patients with simulated data covering a range of conditions considered in the evaluated CGs have been included into the system.

Once the scenario has been configured, the set of CGs described in Table 1 has been modelled. First of all, the correctness of all CGs must be assured. In the case of PROforma, CGs were validated offline using the Tallis Composer in the case of PROforma. SDA Lab4 was used in the case of SDA* CGs.

After that, in a similar way to the case study detailed in Section 7, a step-by-step execution process within the agent-based platform was carried out in order to validate the suitability of the enactment engine and the ontology for several well-differentiated scenarios. Table 2 shows over which patients the CGs were applied according to their corresponding conditions. The scenario has been designed in order to be able to apply every CG to a patient. However, as table shows, in the case of elder patients, several comorbidities were considered. As a result, several CGs were applied one by one. As it is proposed in the future work, in these cases, it is desirable to personalise or adapt general CGs to the several conditions of the patient in order to avoid redundancies in treatments.

HeCaSe2 enacts CGs over patients taking into account their current conditions. In all cases, practitioners monitor the treatment followed, and they can confirm and initiate all required medical actions. Attending the organisational ontology, actors were asked when required, and in most of the

---

3 FaCT++ reasoner is available at http://owl.man.ac.uk/factplusplus/ (last Access January 29, 2011).

Table 1 – Clinical guidelines used for testing.

<table>
<thead>
<tr>
<th>Name</th>
<th>Source</th>
<th>Language</th>
<th>Actions</th>
<th>Decisions</th>
<th>Enquiries</th>
<th>Plans</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prevention of cerebrovascular events</td>
<td>OpenClinical (<a href="http://tinyurl.com/suyx97s">http://tinyurl.com/suyx97s</a>)</td>
<td>PROforma</td>
<td>40</td>
<td>5</td>
<td>32</td>
<td>14</td>
</tr>
<tr>
<td>Prevention of coronary heart disease</td>
<td>K4Care (<a href="http://tinyurl.com/ycmvjt">http://tinyurl.com/ycmvjt</a>)</td>
<td>PROforma</td>
<td>13</td>
<td>11</td>
<td>40</td>
<td>4</td>
</tr>
<tr>
<td>Prevention of exacerbations in stable COPD&amp;CHF</td>
<td>Lozano et al. [47]</td>
<td>PROforma</td>
<td>25</td>
<td>34</td>
<td>13</td>
<td>9</td>
</tr>
<tr>
<td>Prevention of deep vein thrombosis</td>
<td>NGC (<a href="http://tinyurl.com/yka3z6p">http://tinyurl.com/yka3z6p</a>)</td>
<td>PROforma</td>
<td>13</td>
<td>8</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>Management of post-stroke</td>
<td>K4Care (<a href="http://tinyurl.com/ycmvjt">http://tinyurl.com/ycmvjt</a>)</td>
<td>SDA*</td>
<td>10</td>
<td>6</td>
<td>20</td>
<td>1</td>
</tr>
<tr>
<td>Treatment of pressure ulcers</td>
<td>K4Care (<a href="http://tinyurl.com/ycmvjt">http://tinyurl.com/ycmvjt</a>)</td>
<td>SDA*</td>
<td>13</td>
<td>15</td>
<td>12</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 2 – Simulated scenarios.

<table>
<thead>
<tr>
<th>Clinical guideline</th>
<th>Patients</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P1</td>
</tr>
<tr>
<td>Prevention of cerebrovascular events</td>
<td>•</td>
</tr>
<tr>
<td>Prevention of coronary heart disease</td>
<td>•</td>
</tr>
<tr>
<td>Prevention of exacerbations in stable COPD&amp;CHF</td>
<td></td>
</tr>
<tr>
<td>Prevention of deep vein thrombosis</td>
<td></td>
</tr>
<tr>
<td>Management of post-stroke</td>
<td></td>
</tr>
<tr>
<td>Treatment of pressure ulcers</td>
<td></td>
</tr>
</tbody>
</table>

cases, negotiations between different actors were needed to delegate actions. The executions were coherent with respect to patient's data (i.e., decisions taken according to observations, actions performed according to concrete states) and the organisational structure modelled by the ontology (i.e., actions delegated to available practitioners according to their skills and responsibilities).

This shows how the general workflow modelled in the CG can be delivered in different ways depending on the resources and structure of a concrete healthcare organisation, which is represented by means of the organisational ontology.

The use of CGs in daily care permits to standardise the treatments given to patients by providing generic indications. The proposed ontology allows tailoring CGs according to the current circumstances of a healthcare organisation. Thanks to the ontology-driven CG enactment, some repetitive and administrative actions can be automated (e.g., resources allocation, responsible discovery, etc.). As a result, the practitioner is only requested when needed: taking medical decisions, validating or selecting the actions that should be done over a patient, and introducing/modifying data in the EHR.

It is important to note that the inclusion of a new CG in the ontology is a maintenance task, which supposes including (or reusing) medical terms under the Guideline element hierarchy of the ontology and associating them to available Agent types. In this sense, an ontology editor such as Protégé6 (see Fig. 4) can ease this process. Given the backbone structure of classes and relationships presented in Section 6.2, this task does not require taking new ontology engineering decisions but reusing and extending the basic structure with new class specialisations, new ontological restrictions etc. Class inheritance will ensure the coherency of the new specialised classes with respect to the knowledge model. The modelling of different medical organisations or departments (and their associated CGs), however, may require developing separate ontologies (based on the proposed backbone structure) due to significant differences in care delivery and organisational models may exist.

Thanks to the separation of the explicit knowledge embedded into a CG from its execution, CG enactment can be tailored easily. In this sense, the use of an ontology enables introducing necessary modifications to adapt the enactment to changing organisational circumstances with a minimum effort. Concretely, the adaptation or modification of the organisational structure of a medical centre requires defining the internal dependencies between actors, departments and services. As the backbone structure of classes and interrelations is maintained, the multi-agent system is able to process the modelled knowledge homogenously. Thanks to the knowledge driven execution (i.e., procedural knowledge is consulted at runtime according to the executed medical workflows), changes introduced in the ontology immediately affect the behaviour of the multi-agent system. This results in automatic seamless adaptation of the CG enactment to new norms with null re-programming effort and without requiring to re-initiate the systems.

---

6 Protégé website: http://protege.stanford.edu/ (last access January 24, 2011).
9. Conclusions

The translation of narrative documents into computer-interpretable CGs permits the construction of decision support systems but, before adopting CGs in a particular centre, they need to be individually tailored stating the information about how they should be executed. This fact makes difficult their application in a real scenario. In this paper, an architecture aiming to support the enactment of CGs is presented by the inclusion of an especially designed ontology in a knowledge-driven multi-agent system modelling the entities of a healthcare organisation. The ontological paradigm and the expressiveness of modern ontological languages (OWL) allow a proper translation of the organisational structure and medical knowledge involved in the execution of CGs. On the contrary to other approaches in which roles associated to CG tasks are included in each CG (like in SAGE [44]), our approach separates general medical workflows modelled in CGs from its execution in the context of a particular institution as stated in the ontology. In this manner, commonalities between different CGs (e.g., the fact that a practitioner always performs the same action) are only modelled once.

The use of ontologies in the medical domain is increasing and offers some advantages such as making domain assumptions explicit, separating domain knowledge from operational knowledge, and sharing a consistent understanding of what information means. As shown in related works, some researchers tried to represent the whole CG using ontologies [26,28,30], but this kind of systems lack of scalability and generality.

The paper addresses the issue of designing such medico-organisational ontology through the analysis of the knowledge requirements of CG enactment and the use of a standard knowledge engineering methodology. On the contrary to the above-mentioned related works, this ontology is aimed to complement the generic care workflow provided by the CG, instead of replacing it. The designed medical ontology brings the following advantages to the guideline-based execution system: (a) to identify the required actors that are able to accomplish an action, (b) to provide a common and uniform terminology with independence on the CG representation language, easing the incorporation of new CGs by reusing already existing entities, (c) to adapt the execution framework to the particular structure of any healthcare organisation without modifying the system implementation and the guideline either, and (d) to provide an application independent context.

The main goal of the knowledge-driven multi-agent system is to support the medical practitioner during the enactment of a CG in a supervised fashion and to automate the coordination internal activities. In fact, the knowledge-driven design ensures the generality and flexibility of the approach, because the execution behaviour is decided at runtime upon the organisational knowledge stored in the ontology. In this manner, the
system can be easily adapted to changes in the corresponding organisation and actor’s responsibilities simply by changing the ontology.

The backbone design of the ontology eases its maintenance (i.e., adding new CGs) because changes only involve the specialisation/definition of basic or already modelled classes. These changes will inherit properties and constraints that are coherent to the backbone knowledge structure. It is advisable, however, to develop different ontologies for different centres/departments (based on the proposed backbone structure), as the enactment of CGs may significantly change between them.

The exploitation of the agent technology also brings benefits to the coordinated execution of treatments in an inherently distributed setting such as healthcare [39,45]. In fact, coordination between humans and resources is required to execute a guideline. In a centralised model, coordination protocols are difficult to implement or the amount of data to be exchanged may suppose a bottleneck that could hinder system performance. So, the inclusion of a distributed system, as proposed in this work, is a step forward in the development of guideline execution systems.

The proposal, however, can be improved tackling some remaining issues. One of the problems is that the system is designed to deploy general CGs over patients. As shown in Table 2, it is usual that a patient suffers different comorbidities at the same time. In this scenario, some CGs should be merged according the current circumstances of the patient, creating individual treatments, and these treatments should be transmitted among all partners during the enactment in order to assure a correct delivering of care. Following the idea of personalise the delivering of care, as it is proposed in [44,46], it can be interesting to add different roles to the participants in order to describe who must interact with an action and how this interaction can be performed (e.g., establishing permissions). This information can be declared in the CG and interpreted by agents that act autonomously on behalf users.

Conflict of interest statement

The contributing authors have declared no commercial financial conflicts of interest.

Acknowledgements

This work has been supported by the Universitat Rovira i Virgili (post-doctoral grant of D.I., 2009AIRE-04), the Spanish Ministry of Science and Innovation (DAMASK project, Data mining algorithms with semantic knowledge, TIN2009-11005), the Spanish Government (PlanE, Spanish Economy and Employment Stimulation Plan), and the EU-funded K4Care Project (IST-2004-026968).

References


