Review

Computer-based execution of clinical guidelines: A review

David Isern*, Antonio Moreno

University Rovira i Virgili, Department of Computer Science and Mathematics, Intelligent Technologies for Advanced Knowledge Acquisition Research Group, ITAKA, Av Països Catalans, 26, 43007 Tarragona, Catalonia, Spain

ARTICLE INFO

Article history:
Received 31 October 2007
Received in revised form 28 May 2008
Accepted 29 May 2008

Keywords:
Clinical practice guideline
Patient care management
Decision-support systems, Clinical
Decision making,
Computer-assisted

ABSTRACT

Purpose: Clinical guidelines are useful tools to standardize and improve health care. The automation of the guideline execution process is a basic step towards its widespread use in medical centres. This paper presents an analysis and a comparison of eight systems that allow the enactment of clinical guidelines in a (semi) automatic fashion.

Methods: This paper presents a review of the literature (2000–2007) collected from medical databases as well as international conferences in the medical informatics area.

Results: Eight systems containing a guideline execution engine were selected. The language used to represent the guidelines as well as the architecture of these systems were compared. Different aspects have been assessed for each system, such as the integration with external elements or the coordination mechanisms used in the execution of clinical guidelines. Security and terminology issues complement the above study.

Conclusions: Although these systems could be beneficial for clinicians and patients, it is an ongoing research area, and they are not yet fully implemented and integrated into existing careflow management systems and hence used in daily practice in health care institutions.

© 2008 Elsevier Ireland Ltd. All rights reserved.

Contents

1. Introduction ................................................................. 788

2. Guideline execution engines ........................................... 789

2.1. ArezzoTM ................................................................. 790

2.1.1. Clinical guideline representation ................................. 790

2.1.2. Architecture .......................................................... 790

2.2. Digital Electronic Guideline Library (DeGeL) ................. 790

2.2.1. Clinical guideline representation ................................. 791

2.2.2. Architecture .......................................................... 792

2.3. GuideLine Acquisition, Representation and Execution (GLARE) ........................................... 792

2.3.1. Clinical guideline representation ................................. 793

2.3.2. Architecture .......................................................... 794

2.4. GLIF3 Guideline Execution Engine (GLEE) ................. 794

2.4.1. Clinical guideline representation ................................. 794

* Corresponding author. Tel.: +34 977256563; fax: +34 977559710.
E-mail addresses: david.isern@urv.cat (D. Isern), antonio.moreno@urv.cat (A. Moreno).
1386-5056/$ – see front matter © 2008 Elsevier Ireland Ltd. All rights reserved.
doi:10.1016/j.ijmedinf.2008.05.010
1. Introduction

Nowadays, the execution of clinical guidelines (CGs) is one of the most interesting topics of study within clinical informatics. CGs contain a set of directions or principles to assist the health care practitioner with patient care decisions about appropriate diagnostic, therapeutic, or other clinical procedures for specific clinical circumstances [1]. They are intended to ensure consistent high quality clinical practice and provide some benefits to both patients and health care managers [2–8], such as:

- For health care professionals, the use of CGs can improve the quality of clinical decisions and activities and, in consequence, the patient outcomes are also improved (e.g., a clinician will not forget an important aspect to be checked before ordering a certain treatment).
- CGs facilitate reuse of knowledge, because a guideline can be adapted, tailored and applied to different clinical situations.
- Guidelines support rapid dissemination of updates and changes. CGs promote interventions of proved benefits and discourage those that are ineffective.
- CGs help doctors to use the clinical knowledge about the patient at the appropriate point of his care.
- Guideline authors are encouraged to employ rigorous formal techniques, which help to ensure syntactic, logical and medical validity of CGs.

Although several international organisations create and maintain repositories1 with guidelines in different domains (e.g., oncology, cardiology, paediatrics), they are not being widely used in daily practice. Several factors limiting or restricting complete physicians adherence to clinical guidelines were identified in [9]. Such factors include lack of awareness with the guideline’s existence, lack of agreement, lack of physician self-efficacy, lack of outcome expectancy, or the inherent difficulty to change habits in daily behaviour. These factors could be tackled (in most cases) with an automation and computerisation of the daily management of both clinical guidelines and patient data [10].

Several steps must be considered in the management of clinical guidelines: representation, acquisition, verification and execution. The first three tasks concern the authors of the guideline, whereas the later is related to practitioners. Briefly, these steps can be described as follows:

(a) Choice of a representation language. A CG contains several elements to be modelled, such as actions, required patient data, decisions to be made, constraints between

---

1 Good examples of these public repositories are the National Guideline Clearinghouse from the Agency for Healthcare Research and Quality, USA, the Scottish Intercollegiate Guidelines Network developed by the Scottish National Health Service, and the Cochrane Library maintained by the international non-for-profit organisation Cochrane Collaboration.
tasks, temporal constraints in a global plan, etc. Different researchers have defined formal languages to model computer-interpretable clinical guidelines, such as PROforma, EON, GLIF, GUIDE or Asbru [11–14].

(b) Acquisition of CGs. Medical guidelines are based on the evidence collected from clinical trials and existing literature [15,16]. Some authors are also currently working in the semi-automatic construction of guidelines, by applying Machine Learning techniques from a corpus of clinical data collected in a medical centre [17] or directly from textual documents [18].

(c) Verification of CGs. Verification includes two aspects: is a medical guideline well formed?, and, which of these two available medical guidelines is the best? The first question seeks to verify the formal correctness of the guideline [7]. The second question is more difficult to answer since it is necessary to quantify how appropriate is a medical guideline. To tackle this problem, some authors proposed an evaluation procedure called AGREE which calculates a set of parameters for a given medical guideline to evaluate its quality [19]. In addition a methodology to facilitate the whole development and evaluation of clinical guidelines can be found in [20].

(d) Execution aspects. As mentioned above, a medical guideline contains a great amount of information to be considered (decisions to be made, constraints between tasks, temporal restrictions). All these data have to be collected and monitored when enacting the guideline.

Concretely, while representation, acquisition and verification stages are currently active research areas [21–23], the execution of guidelines is a less developed field. This paper is focused on the analysis of systems that allow the automatic (or semi-automatic) execution of guidelines. Several systems are currently being developed for this purpose. In the next section we describe the basic characteristics to be analysed in a guideline execution engine (e.g., coordination issues, security techniques, use of standard medical vocabularies). Section 3 provides a table that summarizes the main information of each system and comments the results obtained for each of the analysed attributes. Section 4 states some comments derived from the comparison of the systems. Finally, the last section briefly summarises a list of concluding remarks.

2. Guideline execution engines

A guideline execution engine should ideally fulfil the following requirements:

- To provide a connection with a computerised patient record.\(^2\)
- To allow the use of standard vocabularies inside guidelines.
- To provide security to both transmissions and storage of sensitive data related to patients.

In the literature, there are other systems related to the execution of guidelines, such as order entry systems and alarm-based systems, which are out of the scope of this paper. Computerised Physician Order Entry (CPOE) systems refer to a variety of computer-based systems for ordering medications and laboratory tests. A basic CPOE system ensures standardized, legible, complete orders by only accepting typed orders in a standard and complete format [25]. With the same goal, there are decision-support systems that can provide advice on drug selection, dosages, and duration [24]. There are computerised systems that monitor certain events and trigger alerts about at-risk states and reminders of appropriate physical assessments and screening activities. Sometimes active systems provide an explanation that offers background information, definitions and risks [24,26,27]. These systems neither consider complex sequences of actions nor coordination aspects, but just react to specific situations.

Before undertaking a deeper analysis of the existing guideline execution tools, we describe the methodology used to select those tools. We searched on the PubMed, SciFinder, ScienceDirect and CiteSeer databases. In addition, proceedings of the most relevant conferences in the domain, such as the Conference on Artificial Intelligence in Medicine, the AMIA Annual Symposium (formerly, the Symposium on Computer-based Applications in Medicine, or SCAMC) and the IEEE Symposium on Computer-based Medical Systems, were also examined. Only relevant articles published between 2000 and 2007 and references therein were considered. The keywords used include: guideline-based execution engine, clinical guideline, computer-interpretable guideline, guideline workflow, guideline execution, and guideline enactment.

All collected papers were analysed and filtered. Eight projects, seven coming from academic research and one commercial system, were selected: Arezzo\(^{TM}\), DeGel, GLARE, GLEE, HeCaSe2, NewGuide, SAGE and SpEM. The main weakness of the followed methodology is that it may leave out industrial or commercial systems, which are usually not described in academic circles (Arezzo\(^{TM}\) being the exception). In contrast, the main strength is that all the systems developed by artificial intelligence (or, more generally, computer science) research groups are certainly covered by this review, and all of them are exhaustively described in research papers.

In the next subsections these tools are succinctly described. The basic features of each tool are summarised, paying special attention to the language used to represent CGs and the proposed architecture.

\(^2\) At the simplest, a computerised patient record is the computer replacement for existing paper medical record systems. It provides mechanisms for capturing information during the medical visit, stores it in a secure fashion, and permits retrieval of that information by those with a clinical need [24].
Arezzo™ is a commercial product to create, visualise and enact PROforma guidelines developed at Cancer Research, UK [28,29].

2.1.1. Clinical guideline representation

Arezzo™ uses the PROforma language to represent CGs [12,30]. PROforma is an executable process modelling language that has been successfully used to build and deploy a range of decision-support systems, guidelines and other clinical applications. It has a declarative format defining four basic types of tasks (plans, decisions, actions and enquiries) as well as logical and temporal relationships between them. An action is a procedure to be carried out (usually by an external element like a doctor or a medical resource). A plan is the basic building block of a clinical guideline and represents a container for a number of tasks, including other plans. A decision is a task that represents an option in terms of different logic commitments to be accomplished. An enquiry is a request for further information or data required before proceeding with the application of the guideline.

2.1.2. Architecture

The tool is composed of three elements: a Composer, a Tester and a Performer (see Fig. 1). The Composer is used to create guidelines using the PROforma language. The Tester is used to test the guideline logic before deployment (it checks that the statements in decisions, tasks and enquiries are well written). The Performer inference engine can then run the guideline, taking into account data related to patients stored in existing health care systems (electronic medical record) [31]. During the enactment of a guideline in the performer engine, a task changes its internal state depending on whether the task is awaiting for data, suspended, finished, or it cannot be accomplished in the current state of the patient.

Arezzo™ uses the Domino autonomous agent model [29]. The model deals with a large class of medical problems and establishes a relationship between decision-making and plan enactment procedures. The main goal of this model is to identify the basic elements required in any language to represent clinical guidelines that can be used for both decision-making and plan management. Fig. 2 shows the whole model, which is divided in two parts: the left-hand side concerns the decision-making processes (steps 1–4), and the right-hand side is related to planning and scheduling of tasks (steps 5–7). The process begins by taking into account a set of patient data. According to the model (step 1) the system proposes a set of possible causes of the health problem (step 2) and identifies a possible set of solutions (step 3) with its associated arguments pro and con. At this point, the doctor can identify a disease that has to be handled (step 4), and the cycle is started again to treat this specific disease. If, at this point, the doctor knows the appropriate option to be followed, he selects the therapy plan (step 5). The component steps of this plan will be scheduled (step 6), resulting in the execution of actions. An action will often produce postconditions that change the patient’s state (step 7). This new information can produce new goals to be managed with other therapies. This is a cyclic model that produces a sequence of decision-making and scheduling steps [12].

2.2. Digital Electronic Guideline Library (DeGeL)

Digital Electronic Guidelines Library (DeGeL) is a web-based, modular and distributed architecture, which facilitates the gradual conversion of clinical guidelines from text to a formal representation in Asbru [32,33]. It is being developed at Ben Gurion University in Israel.

Fig. 1 – Arezzo™ architecture [31].

Fig. 2 – Generalised Domino model [12].
2.2.1. Clinical guideline representation
The system maintains a repository of guidelines, and it allows the user to search, browse, retrieve and visualise all available guidelines. At the moment, the system creates guidelines using the formal language Asbru [34], but the methodology could be extended to other languages.

One of the goals of DeGeL is to create formal guidelines from textual documents. The initial textual guidelines go through an intermediate layer between the textual and the final form, where experts add semantic information. The intermediate layer uses a meta-ontology that defines a hierarchy of basic concepts.
Asbru organises a clinical guideline as a library of Asbru plans created during the decomposition process performed during the specification phase. The Asbru plans in the library are interrelated in a hierarchical network of plans and sub-plans using a parent–child relationship which is encoded using control structures (e.g., do in parallel) [33]. Moreover, two types of plans can be distinguished: atomic and composite. Atomic plans represent a single action to be carried out (e.g., administer a certain drug), whereas composite plans include a collection of atomic or other composite plans.

The system uses different standards to represent the clinical information: LOINC-3 for observations and laboratory tests, ICD-9-CM for diagnosis codes, and CPT-4 for procedure codes [24].

2.2.2. Architecture

DeGeL is a modular system composed of a set of tools that support guideline classification, semantic mark-up, content-sensitive search, browsing, run-time application, and retrospective quality assessment (see the architecture of the system in Fig. 3).

A tool called Uruz allows practitioners or medical experts to create new medical guidelines. Another tool called IndexiGuide facilitates guideline retrieval. The run-time module is composed of several tools to test and visualise CGs. The tool named VisiGuide allows browsing and visualising guidelines. Another element is Vaidurya which allows both searching and retrieving CGs. QualiGuide is a tool that evaluates clinician adherence to clinical guidelines using the intentions of the guideline authors [35]. Finally, Dipole is a tool that assists clinicians to determine a patient eligibility and guideline applicability.

We focus the study in the Spock guideline execution module, which incorporates an inference engine that can retrieve data stored in a patient’s medical record (see Fig. 4). The Spock system is a modular client–server application that consists of: (i) a set of classes, that allow to store any guideline, (ii) a parser, that interprets the content of a guideline, and (iii) a specialized module, the Controller, which synchronizes the communication between the system layers and external services [33,36].

Spock proposes an asynchronous method to monitor all actions made in a guideline and allows to start and resume a CG execution as requested. The method, called application log, stores different data structures, like all state transitions of a plan instance, a queue of scheduled awaiting tasks, and the list of recommended steps issued during application (see Fig. 4).

2.3. GuideLine Acquisition, Representation and Execution (GLARE)

GuideLine Acquisition, Representation and Execution (GLARE) is a system to acquire and execute clinical guidelines, developed at the Computer Science Department of the Università del Piemonte Orientale of Alessandria (Italy) in cooperation with the Computer Science Department of the Università di Bologna (Italy) [39,41].

Fig. 5 – GLARE representation overview [39,41]. (a) Basic entities defined in GLARE and (b) example of CG modelled using GLARE: the gallbladder stones treatment case study.
with Azienda Ospedaliera San Giovanni Battista of Torino (one of the largest hospitals in Italy) [37–40].

2.3.1. Clinical guideline representation
Internally, CGs in GLARE do not use any standard representation. Their authors have defined a proprietary graph-based representation, where each action is represented by a node, while control relations are represented by arcs [41].

The GLARE designers distinguish between atomic and composite actions. Atomic actions are simple actions to be performed in a particular point of the guidelines. Four possible atomic actions were defined: (i) queries, that allow to request any external information, (ii) work actions, that represent actions to be performed, (iii) decision actions, that embed a set of conditions to select an alternative among a set of actions that could be performed at that instant, and (iv) conclusions that allow to describe outputs of a decision primitive. Composite actions are a collection of atomic or other composite actions. For each action there is a set of preconditions, to be fulfilled before its activation, and a set of conclusions, that hold after the execution of the action. The GLARE execution engine maintains the current state of all actions and monitors all their preconditions before starting them. Fig. 5(a) shows the complete taxonomy of actions defined in GLARE as well as a specialisation of queries, work actions and decisions. In addition, Fig. 5(b) depicts a guideline example putting all different primitives together.

Fig. 6 – GLARE general architecture [40].

Fig. 7 – GLEE general architecture [44].
Recently, the authors of GLARE have enriched the internal representation of guidelines in order to support temporal reasoning facilities such as consistency-checking during the creation of a guideline or checking whether actions could be performed a posteriori [42]. This kind of functionalities were implemented by defining a temporal model that manages temporal constraints including constraints on repeated/periodic events [37].

2.3.2. Architecture

GLARE distinguishes between the acquisition phase, when a guideline is introduced in the system (e.g., by a committee of experts), and the execution phase, when a guideline is applied by physicians to a specific situation (i.e. it is instantiated on a given patient).

They define three layers called System, XML and DBMS (see Fig. 6). The System Layer contains the Acquisition and Execution modules. The lower level, called DBMS Layer, connects physically the higher levels with databases where all required data for both creating and executing guidelines are stored. There are data concerning available resources, terminology used in guidelines, information about drugs, information about all open instances of guidelines, a repository of guidelines, and medical records of patients. The intermediate XML Layer allows to exchange data between the DBMS Layer and the System Layer in a structured way [43]. The XML Layer defines an intermediate structure for each database that provides independence between the data and their use.

This system is focused in the management of temporal constraints between different actions in a CG, and the Execution Module allows to execute/simulate a CG using the appropriate retrieved data from each database. Each patient has his own medical record (contained in the Patient DB), which is updated continuously with the actions executed within a CG. The architecture is complemented with a database of available resources in a given hospital (Resource DB), that allows to make domain-dependent execution of guidelines. Moreover, GLARE allows the local adoption and update of guidelines to cope with both the need to apply them to new situations (countries, hospitals and/or departments), and with the need to manage updates (e.g., authoring, recording the history of a guideline and learning from experience) [38].

2.4. GLIF3 Guideline Execution Engine (GLEE)

GLEE is a tool for executing guidelines encoded in the 3rd version of GLIF (called GLIF3), which was developed across different institutions: the Department of Biomedical Informatics (Columbia University, US), the Stanford Medical Informatics Lab. (Stanford University, US), the Decision Systems Group (Brigham and Women’s Hospital, Harvard Medical School, US), the Department of Management Information Systems (University of Haifa, Israel), and Eclipseys Corporation (Boston, US) [44,45].

2.4.1. Clinical guideline representation

GLEE handles guidelines encoded in the GLIF3 language. GLIF3 represents guidelines as flowcharts of temporally ordered nodes called guideline steps that store actions (Action_Steps), decisions (Decision_Steps), and clinical states of the patient (Patient_Clinical_States). There are two more types of nodes, called Branch_Steps and Synchronization_Steps, which are used for modelling multiple concurrent paths through the guideline [46]. Decision criteria are modelled using an OCL-based language (Object Constraint Language) called GELLO [47].

Guideline Execution by Semantic Decomposition of Representation (GESDOR) is an improvement of GLEE, which allows to represent clinical guidelines independently of the chosen representation language [48,49]. The GESDOR model was tested with GLIF3 and a subset of PROforma.

2.4.2. Architecture

As shown in Fig. 7, three levels of abstraction are defined in GLEE: data, business logic and user interface. The data level contains the EMR with a guideline repository and the clinical event monitor, that allows the execution (or simulation) of clinical guidelines through an event-driven model. The business logic level contains the GLEE execution engine, formed by a server and many clients. The server interacts with the data level, and clients interact with users (both through defined interfaces). At the bottom, we find the user interface level, where the clinical applications that exchange data with the upper levels are located.

The execution model of GLEE takes the “system suggests, user controls” approach. A tracing system is used to record an individual patient’s state when a guideline is being applied to that patient. It can also support an event-driven execution model once it is linked to the clinical event monitor in a local environment. The tracing system allows to maintain two main views of the execution. One the one hand, GLEE suggests which actions can be performed and decides which actions (whose preconditions are satisfied) can change the state from started to finished. On the other hand, the user can control the process, and it can initiate, confirm or decide different transitions between actions. Fig. 8 shows the whole process maintained by the GLEE execution engine through the tracing system. Moreover, that tracing system was implemented as an external element and, to facilitate a further analysis (e.g., evaluate the quality of suggestions or audit the sequence of performed actions), it stores all logs using XML (see data level in Fig. 7).

In addition to serving as an interface to the GLIF3 [47] guideline representation model, GLEE defines a collection of public methods (back-end and front-end interfaces shown in Fig. 7) to connect it with electronic medical records and other clinical applications to facilitate its integration with the clinical information system at a local institution. Concretely, two main (widely used) representations were selected to facilitate sharing information across different institutions: resource description framework (RDF) used as language to store the guidelines, and HL7 as generic patient data model [44].

2.5. Health Care Services (HeCaSe2)

Health Care Services release 2 (HeCaSe2) is an agent-based platform [50] that offers health care services to users (patients and doctors) developed as academic prototype at the Intelligent Technologies for Advanced Knowledge Acquisition (ITAKA) Research Group, Universitat Rovira i Virgili, Catalonia.
(Spain). The platform defines an architecture of agents with different roles, and with multiple interactions between them and humans or medical devices [51–54].

2.5.1. Clinical guideline representation
Clinical guidelines in HeCaSe2 are represented using the PROforma representation language [30]. In addition, actions and enquiries are enriched with a medical terminology, the Unified Medical Language System (UMLS [55]), in order to unify the vocabulary and to allow sharing medical knowledge. UMLS is a metathesaurus of terminologies but defines a Code Unique Identifier (CUI) that is used to identify all medical terms [54,56]. Each CUI is labelled with a semantic type, which is used for tailoring its use in the system. In particular, actions can include terms from Diagnostic Procedures, Therapeutic or Preventive Procedures, or a general Health Care Activity; enquiry entries can include Findings, Age groups, Sign or Symptom and other general types.

Internally, the structure of the medical organisation as well as the medical terms are stored in an ontology [54]. The medical ontology provides a common semantic framework to execute CGs, and the involved entities (agents) can coordinate their activities with the appropriate partners. Moreover, ontologies provide a high level abstraction model of the daily workflow. That model can be adapted to each particular organisation, without the agents having to change their internal workflow. In that sense, any organisation can have an ontology adapted to its particular circumstances [56].

2.5.2. Architecture
The authors of HeCaSe2 propose an architecture based on different kinds of agents interconnected following the organisational rules of a medical centre. Each agent acts autonomously with its own knowledge and data. There is not any central control and the number of agents depends on the specific configuration (e.g., doctors, departments, and devices in a medical centre) [57].

The basic architecture of the multi-agent system (MAS) which is being developed in this work is shown in Fig. 9. At the top of the architecture is located the user (a patient), who 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 outsiders accesses to the agents that manage the information of a medical centre. A MCA monitors all of its departments, represented by Department Agents (DAs), and a set of general services linked to human or physical resources, represented by Service Agents (SAs) (e.g., a blood test service). Each department has a staff of several doctors, modelled through Doctor Agents (DRAs), and offers more specific services, also modelled as SAs (e.g., a nurse that can make different observations in situ). 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). In addition, 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). This GA contains only CGs associated to the department where it is located. In addition, the Ontology Agent (OA) provides information about the concepts that appear in a guideline – medical and/or organisational terms – (know-what) and their relations (know-how) [54].

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. Appropriate security measures have been adopted to ensure that only properly authenticated and authorised agents may access and update the medical records (see [58] for more details).

The coordination of the tasks to be performed is made in the DRA by retrieving all information concerned to the patient and following the execution of a guideline applied to him. Using the CG, a doctor can consider all the available data and make an informed decision. During the visit, if the doctor needs another test to be performed on the patient, agents negotiate the best alternative according to the preferences/constraints of the user, the doctor and the hospital services [53]. The medical services needed in the execution of a guideline can be located in the same medical centre or in another one (found with a previous discovery process).
the service is completed, the results are sent automatically from the SA that has made the task to the MRA, that stores all patient's medical records. The interesting point is that all the scheduling processes and the follow-up of a guideline can be made automatically (or semi-automatically, with the doctor checking and validating all relevant details) by agents in an autonomous way, without 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 when the results of a certain patient will be available.

2.6. NewGuide

NewGuide is a framework for modelling and executing clinical practice guidelines developed at the Laboratorio di Informatica Medica, Università di Pavia, Italy [59,60].

2.6.1. Clinical guideline representation

Guidelines are represented using a representation language called GUIDE, which is based on Petri Nets [61]. It allows to model complex concurrent processes as well as temporal, data and hierarchical issues [62].

NewGuide uses the UMLS codification [55] to describe medical terms and procedures (see Section 2.5.1) and a Medical Text Mark-up language called Guideline text Mark-up to describe tasks within a guideline [63].
2.6.2. Architecture

GUIDE is integrated into a workflow management system which proposes an infrastructure that enables inter- and intra-organisational communication through a Careflow Management System (CfMS) that, on the basis of the available best practice medical knowledge, is able to coordinate the care providers activities. The final goal of this architecture is to provide Health Care Organizations (HCOs) with technical solutions which should enable them to improve process efficiency, outcomes and quality of care.

Fig. 10 shows the guideline management system proposed by NewGuide. First of all, there is a division in two levels according to the scope of the CG’s management: central and local. The former is intended to manage guidelines that cover a region, a country or several countries. These (general) guidelines are defined by some health authority or international organisations. The latter (a lower level of coverage) is localised in health care organisations that adopt global guidelines according to their particular requirements. In both cases, the creation and storage of those guidelines follows the same procedure: (i) the CG is created/edited with the NewGuide graphical editor tool, (ii) the guideline repository manager receives that information and splits it into two databases: the general information such as aim, eligibility, author, and version is stored in the first database which is used to search a posteriori, and the rest of the guideline (GUIDE content) is stored in a database of templates.

As shown in Fig. 11, the inference engine is composed of three main elements: a general manager, a message manager, and an instance manager. The inference engine is invoked by a clinician and automatically creates an instance of a CG (previously retrieved according to some criteria) for the management of an individual patient. All the steps followed in the execution of a guideline are supervised by an instance manager. At the same time, all instances are controlled by a general manager. After loading the guideline, the instance manager needs to collect all patient’s data stored in his patient record. The execution engine goes step-by-step recommending actions, such as drug prescription or laboratory tests and, at the same time, stores that information in a logs database. All log data are used to monitor the status of a patient in the CG in another module named reporting system. In addition, the communication between NewGuide and the external world is governed by the message manager, which delegates requests and responses to the web user interface or to an external entity (through a SOAP interface) on the basis of the system configuration. The responsibility for maintaining the correct CG flow and timing is left to the external CfMS [59].

NewGuide authors have studied in detail the concept of non-compliance with guidelines. In [64] they analysed different factors that can cause a doctor not to follow a certain procedure; for example, a guideline does not provide the best recommendations for all patients under all possible circumstances, or a guideline can be applied in different ways depending on the clinical setting. For those reasons, guidelines should be evaluated on the field in order to assess both their applicability and the effectiveness of their implementation. This analysis can be a useful exercise because, according to the type of the detected non-compliance, improvements may be achieved by different interventions, such as site-specification of the guideline, users education, health care administrators involvement, and organisation re-engineering.

2.7. Standards-based Sharable Active Guideline Environment (SAGE)

The SAGE project is a collaboration among research groups at six institutions in the US [65–68]. The project pursues two main goals. First of all, to create an infrastructure that allows medical experts to author and encode guidelines using a standard representation, and then, to use this infrastructure to deploy these guidelines across heterogeneous clinical information systems.

2.7.1. Clinical guideline representation

The internal representation of guidelines in SAGE is made using the EON formalism which is comprised of a set of Protégé classes and plug-ins [69,70].

Fig. 12 shows a portion of a SAGE-defined guideline and how the elements should react to the events in the care process. SAGE defines two different formalisms: recommendation-set and decision-map [66].

The recommendation-set is an activity graph composed of processes and interactions between them. Activity graphs allow the specification of computational algorithms or med-
clinical care plans as processes consisting of (i) contexts, that are combinations of a clinical setting (e.g., outpatient visit in a general internal medicine clinic), care providers to whom the recommendation is directed, relevant patient attributes (e.g., patient age), and possibly a triggering event (e.g., a patient checking into the clinic), (ii) decision nodes, that evaluate conditions on variables (e.g., a Boolean precondition for an action), (iii) action nodes, that encapsulate a set of work items that should be performed either by a computer system or by a health care provider, and (iv) routing nodes, that are used purely for branching and synchronization of multiple concurrent processes. In Fig. 12, C1 and C2 represent context nodes, A1, A2 and A3 are action nodes, D1 and D2 are decision nodes, and R1 is an example of a routing node.

2.7.2. Architecture
The global architecture of the system, that includes a guideline execution engine as its central component, is shown in Fig. 13. It is important to note the integration of this engine with current clinical applications, and also the definition of a central core of terminologies (models that detail not only medical data but also patient data, care workflow processes and the structure of health care organisations)[72].

The execution engine, called SAGEDesktop, is implemented as a centralised element. Given a guideline, it collects the required data from an internal repository and allows medical experts to emulate the real guideline behaviour [65]. As shown at the bottom of Fig. 13, the execution engine interacts with the clinical information system (CIS) via an event listener and a set of services (terminology, patient record and general applications). The terminology server was added to customise the terms used in some specific local applications. Calls to/from the execution engine and the CIS were made through a set of defined APIs, which allow interoperability with existing systems. Services related to the EMR (invoked using medical record calls) allow the engine to retrieve the appropriate patient data. After that, the action service calls allow the engine to initiate actions within the CIS.

The SAGE project proposes a guideline model with the following features:

- It uses standardized components that allow interoperability of guideline execution elements with the standard services provided within vendor clinical information systems. It proposes the use of a repository of CGs in order to manage all available guidelines.
- It uses standards to represent the data (EMR and processes) such as SNOMED-CT and Health Level 7 (in particular, HL7v3) [55].
- It includes organisational knowledge to capture workflow information and the resources needed to provide decision support in enterprise settings. It proposes a methodology to develop/create medical guidelines.

2.8. Specification Execution and Management Plan (SpEM)
Specification Execution and Management Plan (SpEM) is a framework for supporting the management of clinical guidelines [73–75] which was developed at the Department of Computer Science, Dublin Institute of Technology, Dublin, Ireland.

![Fig. 12 – The top-level process specification in a SAGE guideline (adapted from [71]).](image-url)
2.8.1. Clinical guideline representation

The authors of SpEM defined a model that allows to both create and execute clinical guidelines. First of all, a general purpose language called PLAN was adapted to represent clinical guidelines. That language follows an event-condition-action (ECA) approach, based on rules [76]. This ECA rule mechanism is mapped into an existing Database Management System (DBMS), that is at the end who performs the enactment of a specified guideline through rising and managing different triggers [75]. The execution module embedded in a DBMS uses these rules to start a guideline and, according to the raised events, activate an specific task. An ECA rule is composed of three elements: (a) an event part, containing a so-called transition predicate that lists all possible events which are of concern to the rule (it constitutes the situation that the rule has to monitor), (b) a condition part, which can be an arbitrary predicate, and (c) an action part, which is an arbitrary list of executable functions. SpEM defines the following primitives that are required in a guideline or protocol representation model: (a) an action, which represents any clinical or administrative task that is recommended to be performed, maintained, or avoided during the process of guideline application, (b) a decision, that is a selection from a set of alternatives based on predefined criteria in a guideline, (c) a patient state, which is a materialisation of a treated individual’s clinical status based upon the actions that have been performed and the decisions that have been made, and (d) an execution state, which is a description of a guideline’s current state [73].

2.8.2. Architecture

As shown in Fig. 14, SpEM was designed as a layered framework. The highest layer implements the Guideline Management services and it is divided in three main components: Specification plane, Execution plane and Manipulation plane. The first module is able to capture clinical guidelines in a formal way. It provides methods to access, store and manipulate guidelines represented using Protocol Language (PLAN) [73]. The Execution plane provides methods and tools to ease the creation and execution of patient-centred guidelines adapted from guidelines stored in the repository. The last module, the Manipulation plane, provides facilities to query and operate on guideline information through a high level language called TOPSQL [73]. The lower layer implements the active rule extensions through an existing DBMS which supports the ECA mechanism. Between those two layers, an intermediate layer was added to support collaborative features (e.g., manipulation of guidelines between several practitioners) and sharing...
facilities in a generic way (to allow reuse by different applications) [74].

3. Comparison

In the previous section several computer-based systems that manage clinical guidelines have been briefly described. All of them have different scopes, representations and architectures, according to the research interests of each developing group. This section provides a high level comparison of these tools, focused on the items in Table 1. These items are the following:

(a) the existence of a repository of guidelines,
(b) the presence or absence of a tool offering a (graphical) editor to create and visualise its own guidelines,
(c) the formal language used to represent the clinical guidelines,
(d) the basic elements defined in the guideline representation language,
(e) if the tool is designed to be deployed as a distributed system,
(f) the presence of complex coordination elements such as parallelism, negotiation or scheduling,
(g) the type of execution engine (there are different approaches to follow a guideline such as event-based (EB) and rule-based (RB)),
(h) the connection of the system with an electronic medical record (EMR),
(i) the ability to integrate the execution engine with an existing clinical management system (CMS),
(j) the use of any standard terminology or representation language, and finally
(k) the inclusion of security tools to preserve data integrity and authenticate the accesses to the (very sensitive) medical data exchanged in these systems.

3.1. Repository of guidelines

All the tools offer a repository of guidelines. It allows to use the best available guideline at each moment and/or to update them when necessary. In some cases (GLARE, HeCaSe2, NewGuide) the repository stores several versions of a CG, allowing versioning. In particular, DeGeL implements some tools to index the stored guidelines and to enable an automatic search by another decision-support system. This feature can be used to update the CGs or to tailor a general CG to different medical centres according to the resources available in each location. Moreover, HeCaSe2 proposes to distinguish the knowledge available in different departments allowing different repositories for each department in each medical centre. That distribution of the knowledge has several advantages

---

3 The main goal of the EMR is to store and maintain a computerised patient record. It stores all the medical history of the patient with information about results and pending tasks.
4 A CMS is a complex system that includes the management of patients (with access to the EMR) as well as the management of resources, staff, orders and prescriptions.
Table 1 – Comparison of guideline execution engines

<table>
<thead>
<tr>
<th>Tool</th>
<th>CG repository</th>
<th>CG editor</th>
<th>Access to Standards used</th>
<th>Security</th>
<th>Coordination</th>
<th>Run-time engine</th>
<th>Access to CMS</th>
<th>Access to EMR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arezzo™</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>RB</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>DeGeL</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>RB</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>GLARE</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>GLEE</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>RB</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>HeCaSe2</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>RB</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>NewGuide</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SAGE</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>SpEM</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

(e.g., to allow a personalisation of CG to the particular circumstances of the department or the versioning maintained by the doctors of a medical centre), but it hinders the management of all available CGs.

### 3.2. Guideline editor

The use of an editor to create guidelines is a recommended tool to ease the visualisation and updating of guidelines. Most of the described tools offer an editor, which translates the clinical guidelines into the chosen representation language. Moreover, most of them identify several basic components (actions, decisions, and queries) which are linked together using a flowchart-based approach.

Arezzo™, DeGeL, GLARE, GLEE, NewGuide and SAGE have a (graphical) editor to create guidelines in their own representation. Only SpEM and HeCaSe2 lack this module. The former translates CGs directly to database event rules, and the later uses an external tool called Tallis as editor.

Even though a deep analysis of the editors is out of the scope of this manuscript, a good feature that could be implemented in these systems is a web-based edition. In addition, collaborative features to improve the creation of CGs among different medical experts would be very useful.

### 3.3. Language used to represent the computer-interpretable clinical guidelines

There are several available languages and models to represent guidelines [11,13,77]. This is an important drawback because it prevents researchers from implementing tools in the same way, and there is not any de facto standard language. As summarised in Table 1, all platforms define their own language or representation structures, according to their specific goals.

Fortunately, Arezzo™, DeGeL, GLEE, HeCaSe2, NewGuide and SAGE use structured and well-defined languages that can be read, parsed and analysed by a program. That means that these languages can be reused by other organisations in order to implement an ad hoc guideline execution engine or to improve those functionalities (e.g., to edit, to store, to validate) to allow for example a collaborative management.

Guideline representation languages define the declarative knowledge (know-how) of complex medical pathways. The information given in a CG represents the medical knowledge and allows to implement decision-support services at the right time. Although this information is very valuable, process-oriented knowledge (or tacit knowledge) that describes organisation goals, roles and responsibilities, and communication or coordination patterns of the care process, is also required [78,79]. Normally, this knowledge is represented outside of the CG (one of the most used representations is UML) although systems such as NeuGuide include both kinds of knowledge using the same CRM paradigm [61]. Developing guidelines is essentially a consensus process among medical experts. Yet, there is a gap between the information contained in published clinical practice guidelines and the knowledge and information that are necessary to implement them [80–82]. Of the studied CG representation languages, PROforma (used by Arezzo™ and HeCaSe2) is the only approach that makes a distinction between a declarative language (R²L),
used during the guideline acquisition phase, and a procedural language (L2R) that is processed by a general interpreter (PROLOG in this case) in an execution engine [12]. All other approaches require a custom-developed execution engine, in which the different procedural aspects of the guideline are encoded programmatically [11].

3.4. Basic elements defined in the guideline's representation language

As shown in the previous item, there are different languages to represent clinical guidelines, but some of them share common features. Systems such as Arezzo\textsuperscript{TM}, GLARE, HeCaSe2, SAGE and GLEE use similar basic elements (actions, decisions and enquiries). The main difference is related to the management of the links between those elements (e.g., the use of temporal reasoning).

Those languages embed declarative knowledge using different primitives as cycling iterators, logical expressions in decisions, control structures for synchronising sequences of actions, and evaluation of pre- and postconditions before and after the execution of an action.

It is feasible to think that these approaches could converge into a common language to cover all functionalities identified in these representations. Nowadays, the fact that there is no standardisation of representations implies that hospitals implement ad hoc solutions in most of the cases. In this sense, there were several attempts [82-84] to find a common representation format with limited success.

3.5. Agents

Although most of the papers do not use the terms agent or multi-agent system [50], the authors of the analysed systems propose distributed architectures (usually a client-server approach) with both data and tasks deployed around a computer network. In particular, GLARE, GLEE and NewGuide define different types of decentralised systems. Some authors propose an event-driven approach, similar to the communication-based approach, which is the basis of multi-agent systems. Moreover, some of the proposed modules act autonomously, for instance run-time engines, and they could be easily mapped into agents. The HeCaSe2 and Arezzo\textsuperscript{TM} experiences, currently being developed, show how a guideline execution system can be developed using an agent-based perspective with a formal language such as PROforma, which has been designed to interact with external elements linked to agents. This perspective allows to design interoperable platforms that could be extended, in a flexible way, with more services and resources as required in a specific clinical setting. This extensibility and flexibility is especially clear in the HeCaSe2 system.

3.6. Coordination

Clinical guidelines define different tasks to be accomplished. In any run-time engine it is very important to coordinate these tasks efficiently in order to improve the global performance. For instance, some tasks to be performed can have a time constraint (deadline) that should be considered before tackling other tasks; at this point, the system must perform a booking between the patient and the resource that manages the required task. The execution of a CG requires planning, negotiation and scheduling between all entities, resolving conflicts and verifying the consistency of all partial results [85]. In the analysed systems, a run-time engine (central element) controls the execution of the CG. Only SpEM does not show coordination features and the DBMS is who simulates the enactment through the activation of events and rules.

3.7. Run-time engine

A run-time engine is required to simulate the behaviour of a clinical guideline with the patient data values. There are two approaches to perform the simulation: an event-based approach (EB in the table), such as SpEM, GLEE, HeCaSe2 and SAGE, and a rule-based approach (RB), such as NewGuide, GLARE, DeGeL or Arezzo\textsuperscript{TM}.

The difference between event- and rule-based approaches is how the systems are used. The former approach can be used in a continuous system and the events are handled asynchronously as they appear (e.g., the arrival of a patient’s result). The later should be monitored by another partner that supervises and controls the rules that can be activated in any moment in a synchronous way.

3.8. Access to an electronic medical record (EMR)

As reported in [86], there are dozens of implementations of electronic medical records (EMRs) with different functionalities. In all cases a gateway between the guideline execution system and the computerised patient record is needed to retrieve the required patient’s data at each moment. A knowledge base is usually employed to know exactly which attribute is required and to allow the system to find it within the EMR.

Most of the analysed tools implement an interface that enables the communication with a proprietary EMR representation. Usually, the guideline execution engines define a set of (general purpose) interfaces that should be customised for each EMR and provide independence between the data and their use. This integration is the main problem of all guideline-based execution systems. The data definitions required in (formal) guidelines may not map to the data available in an existing EMR, and in practice an extension or modification should be made for every case [81,87]. Recently, Peleg et al. [88] have integrated GLEE with two existing EMR through an ontology-based interface that translates all requirements of the guideline into SQL queries to perform in the EMR.

3.9. Access to an existing clinical management system (CMS)

Extending the previous item, a more general criterion to evaluate is to know whether the systems were designed to be included in an existing clinical management system (CMS) [89]. Normally, these systems provide general functionalities in a health care institution, which could be used by a guideline execution engine. These functionalities mainly comprise the management of information about the available resources of the staff. All the suggestions made by the guideline execution
engine can be used to audit and evaluate general behaviours, identify bottlenecks or dysfunctions between the guideline and the daily practice.

Arezzo™, GLEE, NewGuide and SAGE were designed explicitly with this gateway between those systems (the CMS and the guideline execution). GLARE can also be connected with an existing system but it requires to customize the XML Layer between the system layer and the databases. The rest of the systems were not designed to be embedded into existing systems and, this functionality would require to add new modules or re-design the systems.

Towards this integration, researchers have investigated the way to define and implement interfaces to include clinical decision-support systems into different commercial hospital information systems (HISs). One of the first attempts in this domain was made by Muller et al. [90] who designed two prototypes to include a knowledge-based system into two different commercial HISs. They used XML, CORBA, Java and JDBC calls in order to implement some wrappers to allow getting and putting data into HISs. Another work in this field was made by Schadow et al. [91], who designed a model to integrate guidelines in an electronic health record. In this case, it consists in a HL7-based set of classes embedding logical conditions and goals, that provide an homogeneous representation for both declarative knowledge and data. Finally, Veselý et al. [92] described the integration of a GLIF-based system with an existing information system. In this case, the system was designed as an alerting system but also included a wrapper of the existing system in order to control the changes and get the required data.

3.10. Standards used

Medical terms can change significantly their meaning with little syntactic changes. A solution for that problem is the use of a standard terminology. In [55,87] there is an analysis of a set of currently available approaches, including those used in the tools analysed in this paper. UMLS, which is used in several of the studied systems, links the major international terminologies (e.g., SNOMED, MESH, LOINC and ICD) into a common structure, providing a translation mechanism between them. Each medical term is labelled according to a Code Unique Identifier. Medical terms (CUIs) have a semantic type (e.g., diagnostic procedure, finding, body part), a definition, synonyms, and a collection of relationships with other CUIs (e.g., isA, isPartOf).

The use of a well-known nomenclature is an advantage in order to facilitate the dissemination and automation of the execution of clinical guidelines under any representation. Comparing coding systems is an arduous task and it is context-dependent. Depending on the situation, a codification is more suitable than other and (e.g., UK terms may not perform as well in US-designed systems, or terms in primary care may differ from others used in hospitals). In all these cases, a hand-made human supervision of the used terms should be performed in any guideline acquisition process. When this supervision is made, the use of a codification establishes the basis to interact with any computerised patient record.

In addition, structured languages such as XML or RDF can be used to represent the guidelines internally. This functional-ity facilitates sharing and reusing of existing elements among the different entities involved in the management and creation of guidelines. GLARE, GLEE, DeGeL and SpEM use this kind of representations.

3.11. Security issues

Medical information is sensitive and has to be managed accurately. Transmissions, accesses and storage need a secure handling. To ensure secure transmissions, a ciphering of contents needs to be made [93]. To ensure secure access, only registered (and authenticated) agents have to be permitted to exchange information with any other agent of the system. Finally, data should be stored in a secure database with authentication controls of the agents and users that want to access it. From those analysed systems, only HeCaSe2 implements explicit security features.

In addition, as reported in [86], one of the basic functionalities analysed for a set of EMRs is the provision of log-on/log-off procedures and security issues. The authors of this study identified that 15 of the 40 systems could be qualified as full EMRs.5 A future line of research is to adopt these full EMRs in a guideline execution engine and to take advantage of these facilities.

4. Discussion

This survey has described the basic aspects of several applications oriented towards the automation of clinical guidelines. As a result of this survey, several limitations were identified and should be tackled in the future. One of them is the representation of computer-interpretable guidelines. The representation language is the basis of these tools, and it could be desirable to adopt one formalism as standard and promote the interoperability between different tools and systems. This standardisation seems quite feasible, as most of the representation languages commented in this survey share the same basic components: some kind of action/decision/enquiry nodes, some mechanisms for coordination or synchronisation of actions, the ability to create sub-plans or sub-guidelines (so that different levels of abstraction can be considered when working with guidelines), and the possibility of storing the state of a guideline which is being executed.

Another important element of such automation is the existence of an EMR. The problem of finding the best possible representation of an EMR has not been solved yet, and applications are made ad hoc to fit a certain representation. This also limits heavily the interoperability of different tools.

It is also fair to say that in most countries (even the ones considered to be more technologically advanced) health care is not yet fully computerised, and nowadays it is unrealistic (or it is hard and expensive) to include automated guideline enact-

5 A full EMR is understood as a system that implements: log-on/log-off procedures, security; patient database; charting encounters; prescriptions; chart summary/cumulative patient profile; reports; access to on-line information; referral letters; querying the database, practice research; health maintenance; patient education; laboratory orders and retrieval; and communications and productivity.
ment systems in real clinical settings. Most of the described systems consider as a big concern the seamless integration of the execution of guidelines with the usual workflow of activities within a medical centre, in order to make it feasible to introduce this kind of systems in daily clinical practice. We considered the possibility of including another column in Table 1, reflecting the actual use of each tool in clinical practice, but it seems that none of them is actually in daily use in any medical centre (to the best of our knowledge only Arezzo™ has been used for certain limited tasks).

A common feature in most of the analysed systems is the internal organisation. Basically, we identified three main levels: a level with the patient’s related data, an intermediate level with the execution engine, and a level that contains a set of interfaces to connect the execution engine with external devices. This approach provides transparency between the data and their use and allows to improve each element independently.

Clinical guidelines include sets of rules that a doctor can follow in a specific situation (diagnosis, treatment, or prognosis). Coordination between humans and resources according to these rules is required to follow a guideline in a coherent way (ensuring the satisfaction of all relevant constraints). In a centralised model, coordination protocols are difficult to implement or the amount of data to be exchanged could suppose a bottleneck that could hinder system performance. We think that this is an important issue that shows the appropriateness of distributed systems (e.g., multi-agent systems [50]) for the development of guideline execution systems.

5. Conclusions

This paper analysed and compared different guideline-based execution systems. A brief list of conclusions of this review is the following:

- It is widely accepted that the adoption of guideline execution engines in daily practice would improve the patient care, by standardising the care procedures.
- A guideline stores medical knowledge (declarative) about medical procedures. It is important to use a common vocabulary and adopt one of the available terminologies to permit reuse, learn and share this modelled knowledge.
- The quality of a guideline depends on both its acquisition and its verification. The former has different approaches and there is not any standard or widely used language. The latter is not fully implemented and should be addressed by researchers.
- Guideline-based systems can constitute part of a knowledge-based decision-support system in order to deliver the “right knowledge to the right people in the right form at the right time” [94].
- Guideline execution systems should implement appropriate coordination techniques to perform a complex distributed task (careflow), and to minimise errors and delays between all transitions.
- The inclusion of a guideline-based system or, in a more general way, a clinical decision-support system into an existing (commercial or not) electronic medical record system is hard because they are designed as a closed monolithic system with a lack of interoperability methods.
- This is an ongoing research area with numerous researchers working on it, designing and implementing useful execution systems that could potentially add some benefits to the daily practice.

Acknowledgements

We would like to thank the referees for their valuable comments which helped to improve this paper. The work has been supported by a URV grant, and partially supported by the EU-funded project K4CARE (IST-2004-026968) and the Spanish-funded project HYGIA (TIN2006-15453-C04-01).

Authors’ contributions. The first author selected the eight approaches that were analysed in this review and wrote most of the paper. The second author critically revised several drafts of the paper. Both authors have approved the final version of the manuscript.

References


