Artificial Intelligence in Medicine (2004) 31, 1–27

ELSEVIER

Artificial Intelligence in Medicine

http://www.intl.elsevierhealth.com/journals/aiim

METHODOLOGICAL REVIEW

Approaches for creating computer-interpretable guidelines that facilitate decision support

Paul A. de Clercq[a,b,*], Johannes A. Blom[b], Hendrikus H.M. Korsten[b,c], Arie Hasman[a,b]

[a]*Department of Medical Informatics, University of Maastricht, Maastricht, The Netherlands*
[b]*Division of Signal Processing Systems, Department of Electrical Engineering, Eindhoven University of Technology, Room EH 3.15 POB 513 Eindhoven, 5600 MB, The Netherlands*
[c]*Department of Anesthesiology, Intensive Care and Pain Relief, Catharina Hospital, Eindhoven, The Netherlands*

Received 17 July 2003; received in revised form 26 February 2004; accepted 28 February 2004

KEYWORDS
Computer-interpretable guidelines;
Guideline representation;
Guideline acquisition;
Guideline verification;
Guideline execution

Summary During the last decade, studies have shown the benefits of using clinical guidelines in the practice of medicine. Although the importance of these guidelines is widely recognized, health care organizations typically pay more attention to guideline development than to guideline implementation for routine use in daily care. However, studies have shown that clinicians are often not familiar with written guidelines and do not apply them appropriately during the actual care process.

Implementing guidelines in computer-based decision support systems promises to improve the acceptance and application of guidelines in daily practice because the actions and observations of health care workers are monitored and advice is generated whenever a guideline is not followed. Such implementations are increasingly applied in diverse areas such as policy development, utilization management, education, clinical trials, and workflow facilitation. Many parties are developing computer-based guidelines as well as decision support systems that incorporate these guidelines. This paper reviews generic approaches for developing and implementing computer-based guidelines that facilitate decision support. It addresses guideline representation, acquisition, verification and execution aspects. The paper describes five approaches (the Arden Syntax, GuideLine Interchange Format (GLIF), PRO*forma*, Asbru and EON), after the approaches are compared and discussed.

1. Introduction

1.1. Overview

During the last decade, studies have shown the benefits of using clinical guidelines in the practice of medicine [1]. Utilizing guidelines such as standard care plans, critical pathways and protocols in various clinical settings may lead to a reduction of practice variability and patient care costs, while improving patient care [2]. Although the importance of these guidelines is widely recognized, health care organizations typically pay more attention to guideline development than to guideline implementation for routine use in daily care [3]. Studies have shown that clinicians are often not familiar with written guidelines and do not apply them appropriately during the actual care process [4].

* Corresponding author. Tel.: +31-40-2474794; fax: +31-40-2466508.
E-mail address: p.a.d.clercq@tue.nl (P.A. de Clercq).

0933-3657/$ – see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.artmed.2004.02.003

Implementing guidelines in computer-based decision support systems promises to improve the acceptance and application of guidelines in daily practice because the actions and observations of health care workers are monitored and advice is generated whenever a guideline is not followed. Various studies, covering a wide range of clinical settings and tasks, concluded that the use of these systems significantly improves the quality of care, especially when used in combination with clinical information systems such as electronic patient record (EPR) systems [5]. According to the Institute of Medicine (IOM), these decision support systems are in fact crucial elements in long-term strategies for promoting the use of guidelines [6].

Computer-based clinical guidelines are increasingly applied in diverse areas such as policy development, utilization management, education, clinical trials, and workflow facilitation. Many parties are developing computer-based guidelines as well as decision support systems that incorporate these guidelines [7]. The resulting products show much redundancy and overlap since there is little standardization to facilitate sharing or to enable adaptation to local practice settings [8]. Yet considerable progress has been made and standardized approaches for guideline representation and sharing are central to these efforts [9].

Although many fields contribute to the success of providing guideline-based decision support, often the main focus of researchers is on guideline representation and formalization issues. Two reviews have been published recently focusing on the comparison of computer-interpretable guideline models [10,11].

This paper focuses on active guideline-based decision support systems. Therefore, it not only addresses guideline representation issues, but also addresses aspects concerning (the combination of) guideline acquisition, verification and execution.

1.2. Methods

The approaches discussed in this review are selected on the basis of a literature search and the knowledge of the authors on existing approaches. Inclusion of a paper into the review was based on the following criteria. First of all, as this paper aims at defining and comparing aspects regarding the entire guideline development and implementation process, we selected approaches that focus on different aspects of this process (e.g., guideline representation, acquisition, verification or execution). Other criteria are lifetime and number of publications about the approach.

The literature search was conducted using the 'Medline' search engine, combined with proceedings of the AMIA, MEDINFO and MIE conferences, using the keywords 'guidelines', 'approach', 'decision support', 'representation', 'acquisition' and 'execution' in various combinations.

The final inclusion of an approach as a relevant subject in the review was based on our subjective decision. Therefore, although we recognize that a number of other important approaches exist nowadays such as PRODIGY [12], PatMan [13] and DILEMMA [14], we have limited the number of refereed approaches (also to constrain the size of the review) to the following five: The Arden Syntax [15], GLIF [16], PRO*forma*, [17], Asbru [14] and EON [19].

Another approach that has dealt with the development and implementation of clinical guideline-based decision support systems is the Gaston system [20], which was developed by the authors of this paper. This approach covers most of the characteristics in the areas of guideline representation, acquisition, verification and execution that are already discussed in the five refereed approaches. Therefore it is not discussed in this paper.

The remaining part of this paper defines a number of relevant areas with respect to the guideline development process, after which the selected approaches are discussed and evaluated. The paper finishes with a general comparison of all approaches and a discussion.

2. Areas

We identified four areas that can be distinguished in the process of developing guideline-based decision support systems [10,16,21,22].

- Guideline modeling and representation.
- Guideline acquisition.
- Guideline verification and testing.
- Guideline execution.

For each of these areas, a number of general aspects can be formulated, which will serve as guiding principles in the remaining part of this paper when analyzing the various approaches. The remaining part of this section describes the four areas and their aspects in more detail.

Since great effort must be expended to develop high quality guidelines and in making them computer-interpretable, it is highly desirable to be able to share computer-interpretable guidelines among institutions. Various requirements for sharable clinical guidelines were identified by Boxwala et al. [22].

2.1. Guideline modeling and representation

The question how to represent guidelines is a critical issue. A formal and expressive model should provide (1) an in-depth understanding of the clinical procedures, addressed by the guideline, (2) a precise and unambiguous description of the guideline and (3) a means for automatic parsers to execute guidelines to facilitate decision support. A number of representation-related aspects can be formulated to fulfill the above-mentioned goals.

- *Primitives*: the set of building blocks, used to represent the guidelines (e.g., rules, nodes, frames, etc.) must be expressive enough to capture the various aspects of a guideline.
- *Complexity*: the representation must be able to represent various kinds of guidelines that may differ considerably in complexity and level of abstraction, for example by means of nesting or decomposition.
- *Knowledge types*: guidelines contain a number of different knowledge types such as declarative knowledge (e.g., domain-specific knowledge) and procedural knowledge (e.g., inference or the method of decision support), which should be modeled separately.
- *Didactic and maintenance*: as the content of a guideline is not static but may change over time, the representation must be able store didactic and maintenance information such as author names, versioning information, purposes and detailed explanations.
- *Language*: the representation should be supported by a formal language (vocabulary, syntax and semantics), which has to be expressive enough to capture all the aspects, mentioned in the above points. In addition, a parser must be able to execute the guidelines in order to provide decision support, which requires a syntax that must meet execution-time requirements such as compactness and execution speed.
- *Local adaptation*: as stated above it is important to make guidelines sufficiently general to be shared among different institutions. However, most guidelines undergo changes to make them acceptable to health care providers within a particular setting. These changes must be valid and consistent with the original guideline. When guidelines are updated on an (inter)national level, these changes have to be propagated to the guidelines on an institutional level while keeping the local adaptations intact, which requires sophisticated versioning and adaptation methods.

2.2. Guideline acquisition

An important issue in the development of guidelines is the knowledge acquisition process. Knowledge acquisition tools (KA-Tools) are increasingly used to acquire knowledge directly from a domain expert. These tools may facilitate the knowledge acquisition process by helping domain experts formulate and structure domain knowledge used in guidelines, based on the underlying guideline model. Also, an update mechanism (e.g., version control) must be provided as guidelines may change over time.

2.3. Guideline verification and testing

For acceptance of computer-interpretable guidelines in daily clinical practice, guidelines must be unambiguous and syntactically as well as semantically correct. For example, incorrect advice (e.g., false alarms) is to be kept to a minimum. Verification tests may serve such a purpose. These tests include the detection of various types of logical and procedural errors. In addition, testing guidelines in a simulation environment (e.g., testing the guideline using a number of existing patient records) also increases their validity [23].

2.4. Guideline execution

To provide decision support, guidelines must be encoded in a format, interpretable by automatic parsers that are incorporated in guideline execution engines. Guideline execution engines must be optimized to meet execution-time requirements such as compactness and execution speed. Furthermore, the architecture of the guideline execution engine must be system-independent as well as application-independent so that the guideline engine can be used in multiple clinical domains and in various modes (e.g., proactive versus reactive use) [24].

3. The Arden Syntax

3.1. Introduction

Named after the Arden Homestead conference center, where the initial meeting was held, the first version of the Arden Syntax was developed in 1989 [15] as a response to the inability to share medical knowledge among different institutions. The Arden Syntax (based on the HELP [25] and RMRS [26] systems) is intended as an open standard for the procedural representation and sharing of medical knowledge. It defines a representation for modular

guidelines: Medical Logic Modules (MLMs) [27]. The Arden Syntax focuses on the sharing of 'simple' modular and independent guidelines (e.g., reminders). It is not designed for complex guidelines that for example address treatment protocols [28]. The Arden Syntax was accepted in 1992 as a standard by the American Society for Testing and Materials (ASTM). The current version of the Arden Syntax is Arden 2.0 [29], developed and published by the HL7 group. The Arden Syntax has been used by different institutions and companies to develop and implement guidelines in multiple clinical settings.

3.2. Guideline model and representation

3.2.1.Medical Logic Modules

In the Arden Syntax, each guideline is modeled as a MLM that makes a single decision. Each MLM is an ASCII file, containing slots that are grouped into three categories: *maintenance*, *library* and *knowledge*. The *maintenance* and *library* categories describe the guideline's pragmatics (e.g., title, version, explanation and keywords) and the *knowledge* category describes the logic of a MLM. Fig. 1 shows an example of a MLM that warns a health care provider whenever a patient's hematocrit value becomes too low. The remaining part of this section will explain the various parts of the MLM in more detail.

3.2.2. Maintenance and library slots

As MLMs are to be shared among various institutions, the maintenance and library slots contain necessary documentation for each MLM. As shown in Fig. 1, *maintenance* slots include the MLM's (file)name, author, version, institution, specialist, date of last modification and validation status. The validation status documents whether the MLM has been approved in a certain local institution. This slot may hold the values '*testing*', '*research*' (approved for clinical research), '*production*' (approved for clinical care) and '*expired*' (no longer in use).

When a MLM is shared, the value of the *validation* slot should initially be set to '*testing*', indicating that a receiving institution must approve the MLM for use in clinical care. As MLMs usually require some form of local adaptation before they can be used in a certain institution, changing the value of the *validation* slot to '*production*' implies that the responsibility for the MLM is transferred from the authoring institution to the receiving institution. The name of the person who approves the MLM for local use is stored in the *specialist* slot. As long as a MLM has not been approved for clinical care, the *specialist* slot has no value.

The slots in the library category are used for documentation and consist of the MLM's purpose, a more detailed explanation (which can for example be shown to users when they receive MLM-generated messages) and a number of keywords (for example used to categorize MLMs).

3.2.3. Knowledge slots

The actual medical knowledge is stored into the *knowledge* category. This category consists of five mandatory slots (*type*, *data*, *evoke*, *logic* and *action*) and two optional slots (*priority* and *urgency*). Of these slots, the most important ones are *data*, *evoke*, *logic* and *action*.

3.2.3.1. Data slot

This slot is used to obtain the values of concepts that are mentioned in the MLM from local clinical information systems such as EPRs. For example, the line '*hematocrit:= read last {'hematocrit'};*' indicates that the value of the concept '*Hematocrit*' (used in the logical expression of the MLM in Fig. 1) corresponds to the last hematocrit value in for example an EPR. The terms between the curly braces are often institution-specific: the implementation and integration of the actual interface techniques are usually left to the local institutions [30].

3.2.3.2. Evoke slot

The *evoke* slot specifies the context in which an MLM should be executed. MLMs can be executed as a result of three different types of events: database operations, temporal events and external notifications. The first one is most commonly used. For example, the MLM in Fig. 1 is executed as a result of the '*blood_count_storage*' event (i.e., whenever a new blood count is added to the system's database).

3.2.3.3. Logic slot

The logic slot contains the actual decision criteria that may lead to a certain action. These logical expressions are implemented as production rules and contain concepts that are defined in the *data* slot (e.g., '*Hematocrit*').

The Arden Syntax supports various types of operators such as logical operators (e.g., '*or*', '*and*'), list operators (e.g., '*merge*', '*sort*'), temporal operators ('*after*', '*before*', '*ago*') and aggregation operators ('*sum*', '*average*'). The boolean operators use a three-valued logic, in which the value '*null*' is considered as unknown. Whenever the rule's premise is evaluated '*true*', a particular action that is specified in the *action* slot is carried out. When the premise is evaluated '*false*' or '*null*', the execution of the MLM ends.

```
maintenance:
        title: Alert on low hematocrit;;
        filename: low_hematocrit;;
        version: 1.00;;
        institution: CPMC;;
        author: George Hripcsak, M.D. (hripcsa@cucis.columbia.edu);;
        specialist: ;;
        date: 1993-10-31;;
        validation: testing;;
library:
        purpose: Warn provider of new or worsening anemia.;;
        explanation: Whenever a blood count result is obtained, the hematocrit is checked
                to see whether it is below 30 or at least 5 points below the previous
                value.;;
        keywords: anemia; hematocrit;;
knowledge:
        type: data-driven;;
        data:
                blood_count_storage := event {'complete blood count'};
                hematocrit := read last {'hematocrit'};
                previous_hct := read last ({'hematocrit'}
                    where it occurred before the time of hematocrit);;
        evoke: blood_count_storage;;
        logic:
                /* check that the hematocrit is a valid number */
                if hematocrit is not number then
                        conclude false;
                endif;
                if hematocrit <= previous_hct-5 or hematocrit<30 then
                        conclude true;
                endif;;
        action:
                write "The patient's hematocrit ("|| hematocrit ||") is low or falling
```

Figure 1 An example of a MLM, Arden Syntax keywords are shown in bold [27].

3.2.3.4. Action slot

Once the logical expression evaluates to '*true*', the *action* slot is executed, performing whatever actions are appropriate to the condition. Typical actions include sending a message to a health care provider, adding an interpretation to the patient record, returning a result to a calling MLM, and evoking other MLMs (nesting). For example, the

MLM in Fig. 1 writes a message to the standard destination, stating that the patient's hematocrit value is low or falling (the || operator is a concatenation operator, inserting the actual hematocrit value of the patient into the message). Calling other MLMs is supported in the Arden Syntax by means of the '*call*' statement. Although a MLM can invoke other MLMs, the syntax itself does not support a general control structure to steer these invocations [31].

3.3. Guideline acquisition, verification and testing

Various acquisition tools have been developed to assist guideline authors writing MLMs. Examples include text-based editors where MLMs are typed in as free text, supported by syntax-checkers to improve verification [32] as well as systems that use a controlled vocabulary and 'wizards' to facilitate entering MLMs by unfamiliar users [33,34].

3.4. Guideline execution

In order to execute MLMs, they have to be translated into a format interpretable by a guideline execution engine. A number of implementations for executing MLMs have been developed, including the use of pseudocode [35], C++ [36], Smalltalk and MUMPS. As the Arden Syntax leaves the implementation of patient data modeling entirely up to the local institutions, there are no standard mapping facilities to obtain values of required patient data during guideline execution.

User comments were collected and analyzed over a period of 26 months regarding the system that was in use at the Columbia-Presbyterian Medical Center. The majority of the comments indicated that the messages were actually or at least potentially useful, although a minority indicated that they were unhelpful or actually harmful [37].

4. The GuideLine Interchange Format (GLIF)

4.1. Introduction

The GuideLine Interchange Format was developed to model guidelines in terms of a flowchart that consists of structured scheduling steps, representing clinical actions and decisions. GLIF was developed by the Intermed Collaboratory [38] including researchers at Columbia University, Harvard University and Stanford University and was first published in 1998 [16]. The intended purpose of GLIF is to facilitate sharing of guidelines between various institutions by modeling guidelines in such a manner that the guidelines are understandable by human experts as well as by automatic parsers used in different clinical decision support systems.

Although the first version of GLIF (GLIF2) facilitated the description of more complex guidelines than for example the Arden Syntax did, it still had a number of deficiencies. The model needed improvement in a number of areas. First, important attributes of guideline steps (e.g., criteria) needed to be specified more formally (instead of being described by means of text strings). Also, GLIF2 had no constructs that formally allowed the mapping of (patient) data elements in the guideline onto elements that are used in clinical systems such as Electronic Patient Record (EPR) systems. Furthermore, the number of constructs in GLIF2 was rather limited, constructs that supported for example alternative decisions, iterations, (patient) states, exceptions and events were lacking. These issues have been addressed and have recently resulted in the development of a new version (GLIF version 3) [39], which is discussed in the remaining part of this section.

4.2. Guideline model and representation

4.2.1.Guideline steps

GLIF originated from combining a number of relevant features that were determined from an analysis of the characteristics of a number of existing guideline formalisms: (1) the earlier-mentioned Arden Syntax [15], (2) GEODE-CM, a system that combines guidelines with structured patient data entry and data retrieval from a clinical database [40], (3) MBTA, an architecture for building large knowledge-based medical systems, focused on providing reminders [41] and (4) EON, a component-based architecture for building decision support systems for supporting guideline-based care [19].

The GLIF model is object-oriented and consists of a number of classes that describe typical guideline characteristics (e.g., decisions and actions), attributes of those classes and data types for attribute values. In GLIF version 3, all classes, attributes and relations are described by means of Unified Modeling Language (UML) class diagrams [42]. Fig. 2 shows the main classes that are defined in GLIF version 3.

The *Guideline* object encapsulates a (sub)guideline. This object contains a number of attributes that are administrative in nature (e.g., name and author) but also attributes that describe the capabilities of a guideline (e.g., the guideline's intention and eligibility criteria). A GLIF guideline

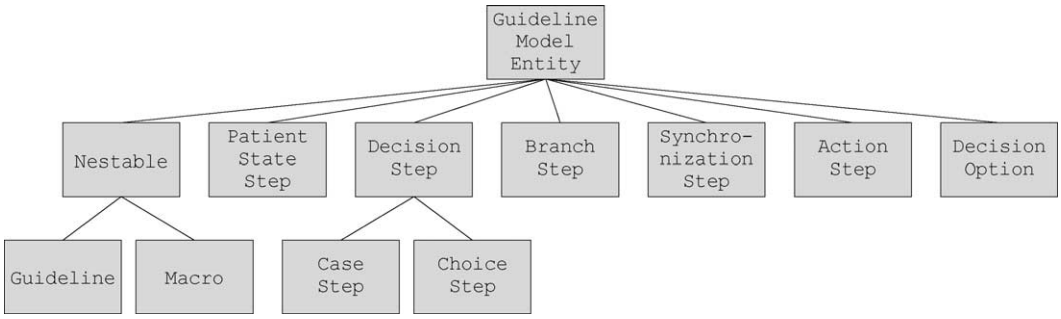

Figure 2 Overview of the main classes in GLIF version 3.

consists of a collection of steps that are linked together in a directed graph (flowchart). GLIF defines five steps: decision steps, patient state steps, branch steps, synchronization steps and action steps.

4.2.1.1. Decision steps
Decision steps model decision points in a guideline and direct flow control from one guideline step to various alternatives. A case step is a decision step that contains a number of logical expressions. Based on the outcome, the guideline flow is directed to the various alternatives. A formal expression syntax (referred to as the Guideline Expression Language or GEL [43] is used), which is a superset of the Arden Syntax.

Another type of decision is the choice step. Choice steps represent situations where a guideline suggests preferences, but leaves the actual choice to an external agent. Choice steps contain rules that support or oppose the various preferences.

4.2.1.2. Patient state steps
A patient state step serves as a label that describes the current patient state that results after having carried out previous steps. It can also be used as an entry point in the guideline, depending on the current patient's state (e.g., the patient revisits a family practitioner with a high blood pressure). Each patient state contains attributes that describe the state of the patient (e.g., the blood pressure is higher than 140/90 during the last week). Whenever this state occurs in practice, the guideline that contains the corresponding Patient state step is executed.

4.2.1.3. Branch and synchronization steps
Branch steps model a set of concurrent steps by directing flow to multiple parallel guideline steps and are used in conjunction with synchronization steps. Multiple guideline steps that follow a branch step always eventually converge in a corresponding synchronization step. When a branch that started at a preceding branch step reaches the corresponding synchronization step, a continuation attribute specifies whether all, some, or one of the preceding steps must have been completed before control can move to the next step. The continuation attribute is expressed as a logical expression.

4.2.1.4. Action steps
Actions steps model tasks that have to (or should) be performed. Three types of tasks are defined: (1) medically oriented actions such as a recommendation for a particular course of treatment, (2) programming-oriented actions such as retrieving data from an electronic patient record or supplying a message to a care provider, and (3) control-oriented actions that invoke nested structures such as (sub)-guidelines or macros to support recursive specification. For example, GLIF defines a MLM-macro, which can be used to define a MLM. Internally, the macro consists of two steps: a decision step and an action step.

4.2.2. Medical ontology
Similar to the Arden Syntax, logical expressions and action specifications in GLIF contain references to actual patient data (e.g., the age of a patient) and clinical concepts (e.g., antibiotic, amoxicillin), which have to be acquired during guideline execution from patient information systems. In order to facilitate sharing of guidelines among different institutions, GLIF aims at defining the structure of these patient data elements and medical concepts in accordance with standard data models and medical terminologies such as HL-7's Reference Information Model (RIM, also known as the Unified Service Action Model or USAM) [44] and the Unified Medical Language System (UMLS) [45]. GLIF divides mapping-related information into three layers: the core GLIF layer, the Reference Information Model (RIM) layer, and the medical knowledge layer.

The first layer, core GLIF, is part of the GLIF language and defines a number of elementary data items and relations that are used as variables in the guidelines. For example, core GLIF defines a *Data_Item* class. Each *Data_Item* class contains a *name*

attribute that specifies the name of the concept. Whenever guideline authors want to use expressions such as 'is amoxicillin being prescribed for more than a week', they refer to a *Data_Item* class instantiation that represents the concept amoxicillin without having to know where this information is stored in a clinical information system.

The medical knowledge layer (third layer) specifies the methods needed to interface with various medical knowledge sources and information systems such as controlled terminologies (e.g., UMLS) and clinical information systems (e.g, EPRs). This layer will contain the information to integrate developed guidelines with institution-specific information systems. However, this layer is still under development.

4.2.3. Guideline representation

Each guideline in GLIF consists of a set of nodes linked together in a temporally sequenced graph (flowchart), in which each node corresponds to an instance of one of the five classes. Fig. 3 shows a graphical GLIF representation of a simple vaccination guideline stating that children under 12 years should receive a pediatric dosage of a certain vaccine whereas health care workers or adults above 65 years should receive an adult dosage.

In order for guidelines to be (1) readable by humans, (2) interpretable by computers and (3) adaptable by different (local) institutions [8], GLIF allows for a specification of a guideline at three levels of abstraction: the conceptual level, the computable level and the implementable level.

The highest level is the conceptual level where guidelines are represented as flowcharts, which can be viewed by humans (e.g., guideline authors) but are not interpretable by decision support systems. At this level, details such as the contents of patient data elements, clinical actions and guideline flow are not formally specified.

These specifications are provided at the computable level. At this level various verification checks of the guidelines are carried out. Finally, at the implementable level, guidelines can be custom-tailored to particular institutional information systems. At this stage, institution-specific procedures and mappings (which are usually non-sharable) are specified, among other things using the above-mentioned medical knowledge layer. Both the medical knowledge and the implementable layer are still under development.

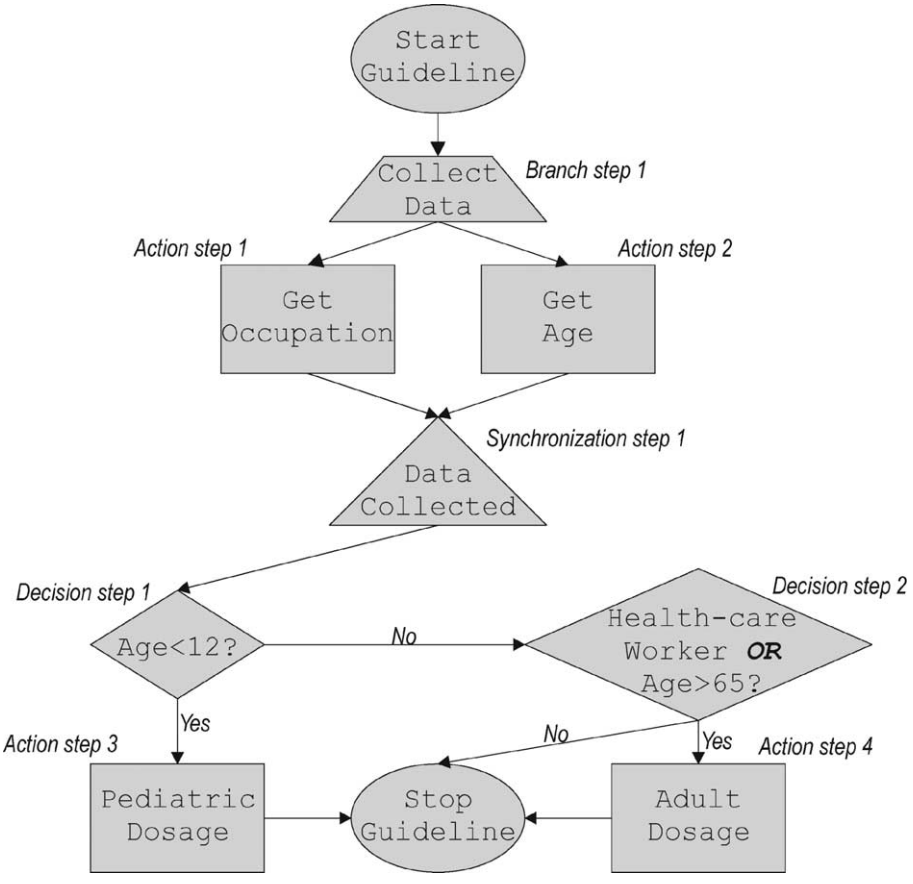

Figure 3 Graphical representation of a guideline in GLIF [16].

```
<a:Guideline rdf:about="&a;Vaccine_INSTANCE_00001">
        <a:name>Guideline for Vaccine X</a:name>
        <a:intention>Decide whether to recommend the Generic vaccine and at what dosage</a:intention>
        <a:algorithm>Vaccine_INSTANCE_00002</a:algorithm>
</a:Guideline>


<a:Algorithm rdf:about="&a;Vaccine_INSTANCE_00002">
        <a:first_step>Vaccine_INSTANCE_00003</a:first_step>
        <a:steps>
                Vaccine_INSTANCE_00003,Vaccine_INSTANCE_00004,Vaccine_INSTANCE_00005,Vaccine_INSTANCE_00006,
                Vaccine_INSTANCE_00007,Vaccine_INSTANCE_00008,Vaccine_INSTANCE_00009,Vaccine_INSTANCE_00010
        </a:steps>
</A:Algorithm>
```

Figure 4 A portion of the vaccine guideline in XML.

4.2.4. Language

In GLIF version 3, an XML-based syntax is used [46] that represents objects and instances of objects in a text-like manner [16]. Fig. 4 shows a small portion of a vaccine guideline in the XML-based syntax.

The upper part shows an instance of the *Guideline* class, identified by Vaccine_INSTANCE_00001, together with values for the name and intention attributes. In addition, the *Guideline* class in GLIF defines an *Algorithm* attribute, which contains a reference to another instance. The latter contains references to all steps present in the guideline, including a reference to the first step.

Recently, the object-oriented expression language GELLO was proposed to specify decision and eligibility criteria in GLIF [47]. In contrast with the original GEL language, the GELLO language is able to include references to concepts and attributes from the core GLIF model.

The GLIF model, representation and syntax are still under development. A variety of guidelines [48—50] are being specified to evaluate the various aspects of GLIF.

4.2.5. Modeling tools

Currently, two tools are used to develop the GLIF model in terms of classes and attributes: Protégé [51] and GEODE [52]. These tools are also used for creating the relations between the core GLIF items and the concepts from the RIM and medical knowledge layer.

4.3. Guideline acquisition, verification and testing

Besides model development tools, Protégé and GEODE are also used as knowledge acquisition tools to facilitate the entering of guidelines. Both tools visualize GLIF guidelines by means of flowcharts. For example, Fig. 5 shows part of a cough treatment guideline entered by means of the Protégé knowledge acquisition tool.

The left pane shows a graphical overview of the guideline in terms of a flowchart. The right pane shows an overview of all available guideline steps (e.g., *Action step*, *Patient state step*, etc). All objects in the right pane can be selected and dropped onto the left pane, thus creating the flowchart (level A). Selecting a step in the flowchart brings up a form in which the details of that step can be filled in (level B). For example, in Fig. 5, the *Cough gone* step has been selected which is a *Choice step*. As a result, the various alternatives can be specified (these are stored in the *Options* attribute).

The use of a RIM combined with an expression syntax allows performing verification tests, as patient data elements, logical criteria and control flow are formally defined. Recently, Peleg et al. [53] described a tool for validating the syntax of guidelines, data type matches, cardinality constraints and structural integrity constraints.

4.4. Guideline execution

For GLIF2, efforts have been undertaken to develop guideline execution engines such as the Partners Computerized Algorithm Processor and Editor (P-CAPE) tool [54]. As GLIF version 3 is still under development (especially the medical ontology layer and the implementable level), GLIF version 3 guideline execution engines are also still under development. The most recent development is the GuideLine Execution Engine (GLEE), which is able to execute GLIF-encoded guidelines and can be integrated into the clinical information system of a local institution [55].

5. PRO*forma*

5.1. Introduction

PRO*forma* is a knowledge composition language supported by acquisition and execution tools with the goal of supporting guideline dissemination in the form of expert systems that assist patient care through active decision support and workflow

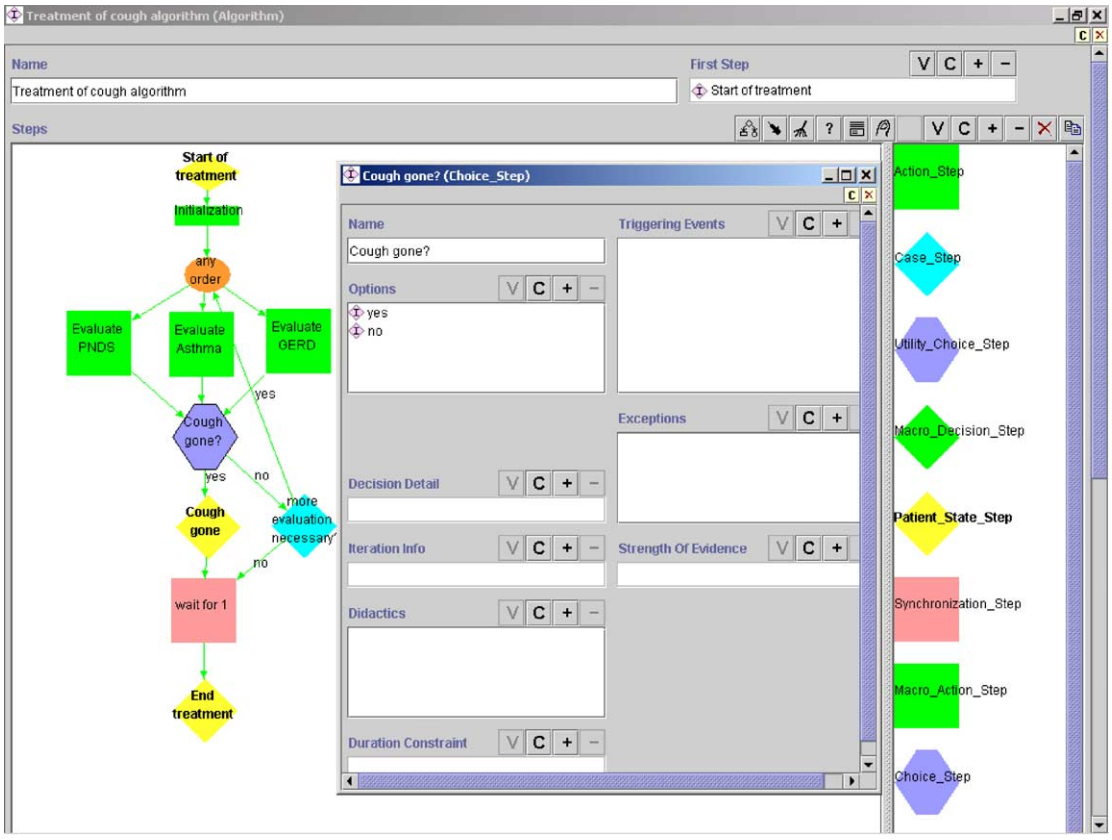

Figure 5 Part of a GLIF cough treatment guideline as a flowchart in Protégé.

management [17]. PRO*forma* was developed at the Imperial Cancer Research Fund by John Fox and colleagues and aims at the development of reliable expert systems that assist patient care through active decision support and workflow management. The name PRO*forma* is a concatenation of the terms proxy ('authorized to act for another') and formalize ('give definite form to').

5.2. Guideline model and representation

5.2.1.The task ontology

Similar to the GLIF model, PRO*forma* also represents guidelines as a directed graph in which the nodes are instances of a closed set of classes, called the PRO*forma* task ontology. Each guideline in PRO*forma* is modeled as a plan that consists of a sequence of tasks. The PRO*forma* task ontology defines four task classes, each with their own attributes: (1) plans (2) decisions, (3) actions and (4) enquiries (Fig. 6).

5.2.1.1. Root task

All tasks are derived from the root task. The root task contains a number of attributes that are common to all four derived tasks. These include administrative ones that hold a name, caption or description but also attributes that describe the capabilities of a task such as goals (e.g., '*achieve(normal_respiration)*'), pre- and postconditions (e.g., '*risk_level=severe*'), trigger conditions (e.g., '*peak_flow < 30*') and cycles (e.g., '*cycle(integer, interval)*').

5.2.1.2. Plans

Each plan models a (sub)guideline. Plans define (1) an ordered sequence of tasks, (2) logical and temporal constraints on their enactment and (3) circumstances in which a plan must be aborted or terminated (e.g., exceptions). Besides the common attributes that are defined in the root task, the plan task contains additional attributes such as *Components*, *Scheduling* and *Temporal constraints* and *Abort* or *Termination conditions*. The *Components*

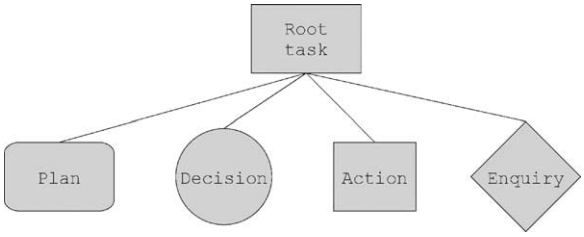

Figure 6 The PRO*forma* task ontology.

attribute is a container that holds a set of task instances, similar to the *Steps* attribute of the *Algorithm* class in GLIF. For example, a guideline that consists of 4 task instances (e.g., '*history*', '*diagnosis*', '*therapy*', '*follow-up*') is modeled through a *Plan* instance of which the *Components* attribute contains references to those four task instances.

The ordering between these task instances is defined by means of two sorts of constraints: scheduling constraints and temporal constraints. Scheduling constraints order tasks in a plan by means of qualitative conditions (e.g., the '*history*' task is executed '*before*' the '*diagnosis*' task). Temporal constraints order tasks by using temporal conditions (e.g., the '*follow-up*' task is executed '*after a period of 10 weeks*'). By using these two types of constraints, tasks in a plan are not modeled as traditional flowcharts that order guideline elements usually only through scheduling constraints.

Another way of directing guideline flow in PROforma is through abort or termination conditions. Each PROforma task passes through a number of states such as '*dormant*', '*in progress*', '*aborted*', '*terminated*' and '*performed*'. Every task is initially in a '*dormant*' state. Executing a certain task changes its state from '*dormant*' to '*in progress*'. Whenever a task is finished normally, the task's state becomes '*performed*'. It is possible to force the termination or abortion of a plan by means of the abort and termination conditions.

5.2.1.3. Decisions

A decision task is represented as a set of possible outcome candidates plus various types of schemas (logical expressions) that support or oppose each candidate. Every candidate is associated with a set of schemas. Schemas consist of rules, qualitative variables, quantitative weightings and certainty factors [56] and support (+) or oppose (−) candidates, establishing a preference order among them. For example, the fact that a patient is diagnosed with oesophagitis, combined with the fact that (s)he has no liver disease supports the prescription of cimetidine. This can be translated into an argument schema: '*diagnosis = oesophagitis and liver_disease = absent then cimetidine: +*'. Besides schemas, decisions also include mandatory data constraints. These state that certain data (e.g., '*presence of liver_disease*') has to be available before a decision can be taken.

5.2.1.4. Actions

An action is a task that a PROforma execution engine can request for enactment by an external agent (e.g., a clinical user or an external software program or hardware device). Such an action in PROforma usually exists of issuing a message to a user or calling an external program through a predefined Application Programming Interface (API). Examples are '''*give ibuprofen*, 10 mg''' that shows a message to a clinical user or '*call(print(leaflet1)')* that executes an external procedure to print a leaflet. In PROforma, actions are always atomic and are not decomposable.

5.2.1.5. Enquiries

Enquiries are used to acquire various kinds of information, such as clinical or administrative information. This information can be obtained from a clinical user or can be directly extracted from an external software agent or hardware device (e.g., EPR or patient monitor). Therefore, as was the case with the definition of an action, the *Enquiry* class contains attributes that define the method of data retrieval.

5.2.2. Guideline representation

Fig. 7 shows an example of a guideline in terms of instances of plans (rounded rectangles), decisions (circles), actions (square rectangles) and enquiries (diamonds).

PROforma contains temporal as well as scheduling constraints. Therefore, the arrows in Fig. 7 may represent both these constraints and merely state that there is some kind of relationship between linked concepts.

5.2.3. Language

Guidelines in PROforma are stored (in terms of instances of task classes) using the Red Representation Language (R^2L), a time-oriented knowledge representation language [57]. A guideline, written in R^2L, is a declarative specification of tasks and their (inter)relationships organized in a hierarchy of plans and their components. An example of a PROforma guideline in R^2L is shown in Fig. 8.

Before execution, guidelines in the R^2L language are translated into another language, called L_{R2L} ('Logic of R^2L'), a language based on predicate logic. This language is used as input for the verification and execution modules (explained in the next sections).

5.3. Guideline acquisition

PROforma contains a number of tools to develop guidelines [58]. The PROforma task authoring environment enables guideline authors to define guidelines in terms of class instances (and attributes) of the task ontology. Fig. 7 shows a part of a treatment protocol that has been entered in the task authoring environment.

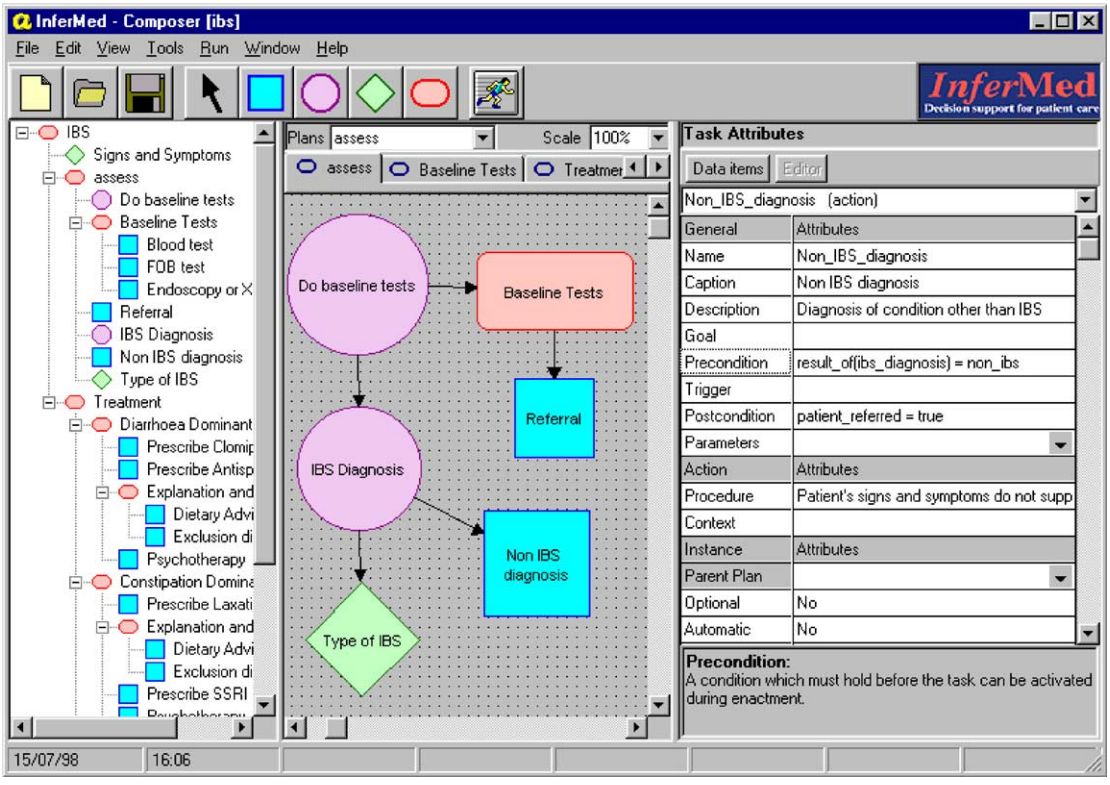

Figure 7 A part of a guideline, entered in the task authoring environment.

The left pane of the task authoring environment shows a treelike overview of all (sub)plans that a guideline contains, whereas the middle pane shows a graphical representation of the currently selected plan.

When a task is selected in the tree, the right pane shows its attributes. Entering guidelines in PROforma is a two-phased process. First, a graphical layout of the plan is specified in terms of instances of the four tasks, without entering attribute-specific values. The latter is done in the second phase where for each instance its attributes are filled in.

5.4. Guideline verification and testing

A major focus point of the PROforma approach is to increase the safety of guidelines. Unsafe situations may occur as a result of incorrect or incomplete knowledge as well as incorrect or incomplete reasoning strategies. In order to address these problems, the PROforma researchers developed a life cycle for the engineering of knowledge base systems [59]. In this lifecycle, guidelines that are acquired by means of the PROforma task authoring environment are stored in the R^2L language, after which they are processed by a verification tool to detect errors that are declarative in nature such as incorrect data types, invalid syntax, missing task values (e.g., missing candidates or decision rules), inconsistent data references and inconsistent scheduling or temporal constraints.

Guidelines are then translated into the L_{R2L} language. The guidelines in L_{R2L} are then processed by a PROLOG-like interpreter in the PROforma execution engine, which is embedded in a test environment. In this environment, users are able to view and evaluate guidelines (an example of the user interface of the execution engine is shown in Fig. 9). After a certain test period, guidelines can be updated through the task authoring environment or transferred to the execution engine used in daily practice.

PROforma also defines an extension of the L_{R2L} language, called L_{safe}, which defines additional safety-related operators such as integrity and safety constraints [60].

5.5. Guideline execution

As mentioned in the previous section, the PROforma framework also contains a standard execution engine that executes entered guidelines by parsing and interpreting a L_{R2L} task definition. The execution engine is able to directly read and execute guidelines and can be interfaced through an API to various interfaces. Fig. 9 shows an example of the PROforma execution engine user interface. The engine executes tasks according to a control regime in which tasks pass through a sequence of states

```
plan :: Protocol1 ;
        caption :: 'Management of weight loss (simplified)' ;
        precondition :: problem = weight_loss ;
        goal :: clinical_goal = manage : weight_loss ;
        component :: enquiry1 ;
        Component :: decision1 ;
                schedule_constraint:: completed(enquiry1) ;
        Component :: decision2 ;
                schedule_constraint:: completed(decision1) ;
        component :: plan1 ;
                schedule_constraint :: completed(decision2) ;
        Component :: plan2 ;
                schedule_constraint :: completed(decision2) ;
        component :: plan3 ;
                schedule_constraint :: completed(decision1) ;
end plan .

decision :: decision1 ;
        caption :: 'Diagnosis?' ;
        goal :: goal = manage : cancer ;
        source ::
                age; mandatory :: yes  ;
                smoker; mandatory:: yes ;
                biopsy; mandatory :: yes ;
                pain: site; mandatory :: yes ;
                pain: time; mandatory :: yes ;
        choice_mode :: single ;
        support_mode :: symbolic ;
        candidate :: cancer ;
                argument :: ( age = elderly) + ;
                argument :: ( smoker = yes) + ;
                argument :: ( biopsy = positive) + ;
                argument :: ( pain: time = immediate) + ;
                argument :: ( pain: site = epigastric) + ;
                recommendation ::
                        netsupport( decision1, cancer) >= 1 ;
        candidate :: peptic_ulcer ;
                argument :: ( age = young or age = adult) + ;
                argument :: ( biopsy = negative) + ;
                argument :: ( pain: site = epigastric) + ;
                argument :: ( pain: time = delayed) + ;
                recommendation ::
                        netsupport( decision1, peptic_ulcer) >= 1 ;
end decision .
```

Figure 8 A part of a guideline in R^2L [57].

(e.g., '*in progress*', '*terminated*' or '*abandoned*'), in which the sequence is determined by situations that are encountered by the system. When required, the engine collects information (e.g., from clinical users or external devices) and takes actions (e.g., sending a message).

The left pane shows an overview of all tasks and their current state, indicated by different colors of the task's icon. For example, the icon of the '*signs_and_symptoms*' enquiry is blue, which indicates that the task is '*performed*'.

The right pane shows a more detailed overview of the currently executed task. In this case, the '*signs_and_symptoms*' enquiry task has been completed, resulting in the execution of the '*do_baseline_tests*' decision task. Based on already known patient data (for example the patient's age, which is already filled in by the user), two possible candidates ('*yes*' or '*no*') are shown, of which '*no*' is recommended by the system, based on the currently evaluated schemas of this particular decision task. Before continuation, the user first has to commit to the decision.

Various decision support systems were developed and implemented using the PRO*forma* approach, some of which are currently in clinical use [61]. Also, a commercial version of PRO*forma*, named Arezzo, has been developed by *Infer*Med Ltd. Examples of developed decision support systems can be found there [62].

6. Asbru

6.1. Introduction

Asbru is a guideline representation formalism, developed at Stanford University and the Vienna

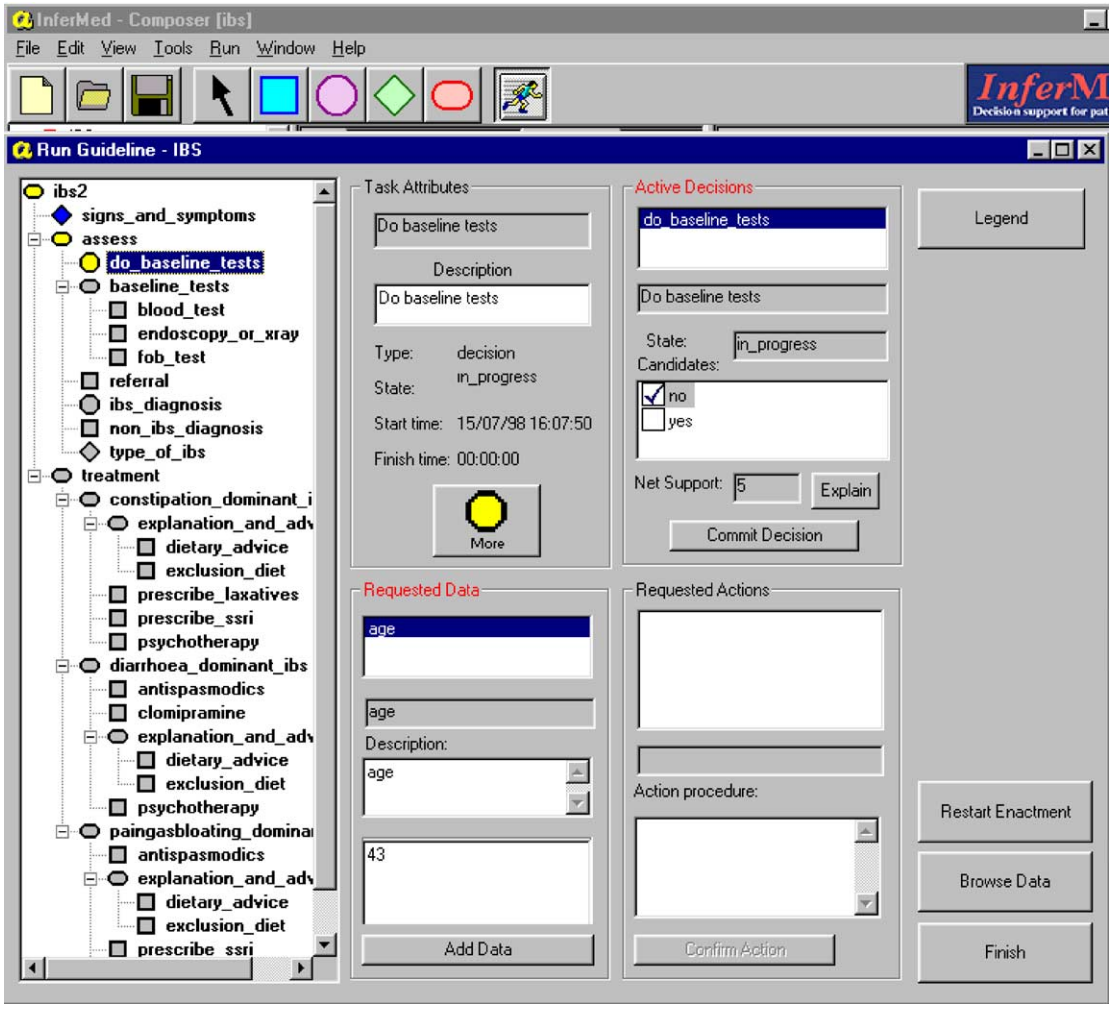

Figure 9 Execution of a guideline in the PRO*forma* execution engine.

University of Technology and is part of the Asgaard project [18], which focuses on the application and critiquing of time-oriented clinical guidelines. The Asbru language [63] is a plan representation language that represents clinical guidelines as time-oriented skeletal plans, which are plan schemata at various levels of detail. In order to manage these (often complex) skeletal plans, key aspects of Asbru are the representation of high-level goals (intentions), the representation of temporal patterns and time annotations, and the development of user interfaces to visualize developed plans.

6.2. Guideline model and representation

6.2.1.The intention-based model

Asbru uses an intention-based model to represent clinical guidelines as skeletal plans. Similar to the notion of plans in PRO*forma*, a plan is a collection of other items. The Asbru model identifies a number of general tasks, which have to be carried out during the process of acquiring, testing and executing guidelines. Examples of these tasks are guideline verification and validation, applicability, execution, recognition and critiquing. Each task is performed by means of problem-solving methods (PSMs), which are generic strategies to solve stereotypical tasks, independent of the system's application domain [64]. The knowledge needed to solve a certain task is defined by means of knowledge roles, which give an abstract description of the function domain knowledge has to play in a PSM. Knowledge roles are specified by a guideline author during guideline acquisition. The Asbru language introduces the following knowledge roles: preferences, plan intentions, conditions, effects and a plan body. In Asbru, the content of a plan (the plan body) always consists of other plans, until a plan is no longer decomposable. The latter is referred to as an action. In Asbru, guidelines entirely consist of plans and actions. The functionality of each plan is modeled by means of a number of knowledge roles. This,

in contrast to other approaches where the functionality of a guideline is described in terms of its primitives. Asbru defines the following knowledge roles, which are part of each plan.

6.2.1.1. Preferences
Preferences bias or constrain the applicability of a plan to achieve a certain goal. Examples of preferences are (1) '*select-method*', a matching heuristic to determine the applicability of the entire plan (e.g., '*exact-fit*' or '*roughly-fit*'), (2) '*resources*', a specification of forbidden or obligatory resources (e.g., in certain cases of a pulmonary infection treatment, surgery is prohibited and antibiotics must be used), and (3) the applied '*strategy*' (e.g., '*aggressive*' or '*normal*').

6.2.1.2. Intentions
One of the key aspects of Asbru is the representation of intentions of a plan: high level goals at various levels of a plan. Besides aiding in the selection of the most appropriate plan, intensions are primarily used in the process of providing decision support. For example, in a guideline for the treatment of hypertension, one possible course of action may be the prescription of beta-blockers in order to lower the blood pressure. However, it is possible that a physician for some reason decides not to use beta-blockers, but aims at lowering the blood pressure in another way. Although the physician follows the plan's intentions, (s)he technically does not follow it, so a guideline execution program that monitors the physician's actions may critique the physician that (s)he is not following the plan. However, if the guideline execution program recognizes from the plan's intentions that its goal has been reached it will not generate a critique, which will improve the acceptance of the system. Intentions are defined as temporal patterns of provider action and patient states that must be maintained, achieved or avoided. Four categories of intentions are defined.

1. *Intermediate state*: the patient states that must be maintained, achieved or avoided (e.g., weight gain levels of slightly low to slightly high).
2. *Intermediate action*: the provider actions that should take place during the execution of the plan (e.g., monitor blood glucose once a day).
3. *Overall state pattern*: the overall pattern of a patient state that should hold after finishing the plan (e.g., patient has an adequate glucose level).
4. *Overall action pattern*: the overall pattern of provider actions that should hold after finishing the plan (e.g., patient has visited a dietician regularly for at least 3 months).

6.2.1.3. Conditions
Conditions are also temporal patterns and are used to change the state of a plan. In Asbru, similar to the PRO*forma* approach, plans are in a certain state during execution-time (e.g., '*activated*', '*suspended*', '*aborted*' and '*completed*'). Asbru defined a number of condition categories such as '*filter-preconditions*' and '*setup-preconditions*' that need to hold if a plan is considered applicable, '*suspend-conditions*' that determine when an active plan must be (temporarily) suspended, '*abort-conditions*' that determine when an active or suspended plan has to be aborted and '*completed-conditions*' that determine when a plan is (successfully or not) completed.

6.2.1.4. Effects
Effects describe the relationship between plan arguments and measurable effects by means of mathematical functions (e.g., the insulin dose is inversely related in some manner to the level of blood glucose). Effects may include probabilities that specify the probability of the effect's occurrence.

6.2.1.5. Plan body
The plan body is a set of actions or plans that have to be performed whenever the preconditions hold. A plan is composed of other plans, which are performed according to the plan's type. Asbru defines three plan types: '*sequential*', '*concurrent*' and '*cyclical*', the aspects of which are described by means of the *plan'subtype*' attribute. An examples of a subtype is '*DO-ALL-TOGETHER*' indicating that all plans in the plan body must be completed concurrently. Each plan is decomposed into subplans until a non-decomposable plan (called an action) is encountered.

6.2.2. Temporal patterns and time annotations
Important in Asbru are time annotations: specifying temporal aspects of a plan. A time annotation specifies four points in time relative to a reference point, which can be a specific or abstract point in time, or a plan's state transition. These four points are: the earliest starting shift (ESS), latest starting shift (LSS), earliest finishing shift (EFS) and latest finishing shift (LFS). Two durations can also be defined: The minimum duration (MinDu) and maximum duration (MaxDu). Together, these data specify the temporal constraints within which an action must take place, or a condition must be fulfilled in

order to trigger [18]. The Asbru temporal representation also supports the concept of temporal abstractions, in which guideline authors are able to specify expressions such as 'has the patient suffered from a *second* episode of anemia of *at least moderate severity*'.

6.2.3. Guideline representation

In Asbru, a guideline is represented by means of a plan, which consists in turn of a collection of other subplans. Plans are executed sequentially or in parallel. As mentioned earlier, plans that have been started can be suspended, aborted or completed (based on the plan's conditions). When a plan is completed, the next plan in the sequence (if any) is executed (only one plan at a time can be activated). Fig. 10 shows a portion of a guideline for the treatment of Infants' Respiratory Distress Syndrome (I-RDS), encoded in Asbru.

This guideline consists of 4 plans that are executed sequentially. The most important plan ('*one-of-controlled-ventilation*') consists internally of three subplans ('*controlled-ventilation*', '*permissive-hypercapnia*' and '*crisis-management*'), which are also executed sequentially, although in this case the order of the sequence depends on the outcome of the '*one-of-controlled ventilation*' plan's '*continuation-conditions*' (which specify the severity of the I-RDS disease).

6.2.4. Language

The formal syntax of the Asbru language is defined in Backus—Naur Form (BNF) [63]. The guidelines are encoded in a LISP-like language, as shown in Fig. 10.

The first paragraph shows the main I-RDS guideline, which contains four subplans that should be executed sequentially. As mentioned earlier, the '*one-of-controlled-ventilation*' plan consists of three subplans that are sequentially executed in some order before continuing (shown partly in the second paragraph). The third paragraph shows one of these subplans ('*controlled-ventilation*') in more detail. The aim of this plan is to maintain a normal level of the blood-gas values and the lowest level of mechanical ventilation (as defined in the context of controlled ventilation therapy). The plan is activated when the Peak Inspiratory Pressure (PIP) is smaller than or equal to 30 and the transcutaneously assessed blood-gas values are available for at least 1 min after activating the last plan instance initial-phase. The plan must be aborted when the PIP is greater then 30 or the increase of the blood-gas values is too steep for at least 30 s. Every 10 s, the abort conditions are evaluated. The plan is completed successfully when the FiO_2 is smaller than or equal to 50%, the PIP is smaller than or equal to 23, the breathing frequency is smaller than or equal to 60, the patient is not dyspnoeic, and the blood-gas values are normal or above the normal range for at least 3 h. The complete conditions are evaluated every 10 min. The body of the plan again consists of two subplans ('*one-of-increase-decrease-ventilation*' and '*observing*') that are executed sequentially.

Besides the BNF-based and LISP-like syntaxes, an XML-based version of the Asbru syntax was also recently defined and published [65].

6.3. Guideline acquisition, verification and testing

In contrast to approaches such as GLIF and PROforma, the developers of Asbru have chosen not to visualize guidelines by means of a flowchart, mainly as they feel that visualizing time and intentions through flowcharts is a very difficult task. Instead, a tool named AsbruView was created that uses the concept of metaphor graphics to visualize guidelines [66]. In AsbruView, plans are visualized as running tracks and the various types of conditions are visualized by means of traffic signs and other controls. This visualization is known as the topological view. Fig. 11 shows part of the I-RDS guideline in the AsbruView topological view.

The length of the track represents time, the depth represents (sub)plans that are on the same level of decomposition and the height represents the various levels of decomposition. In this case, the main guideline ('*I-RDS-Therapy*') contains two sequential subplans: '*initial-phase*' and '*one-of-controlled-ventilation*' (The '*weaning*' and '*one-of-CPAP-extubation*' plans in Fig. 11 are omitted here). The '*initial-phase*' plan also contains two sequential subplans ('*set-respirator-settings*' and '*observing-blood-gas*') and the '*one-of-controlled-ventilation*' subplan contains the four earlier-mentioned subplans (see also Fig. 10). The time dimension is only symbolic: a plan's size does not reflect its actual duration.

Furthermore, AsbruView uses other metaphors to symbolize conditions. For example, the 'no entrance with exceptions' traffic sign symbolizes the filter preconditions and the turnpike (barrier) sign symbolizes setup preconditions. Furthermore, each traffic light includes three kinds of conditions. The red light symbolizes the abort-condition, the yellow light the suspend-condition and the green light the reactivate-condition. The finishing flag, finally, symbolizes the complete condition, which specifies when the plan has reached its goal and can be considered successful.

```
(PLAN I-RDS-therapy ...
  ...
  (DO-ALL-SEQUENTIALLY
    (initial-phase)
    (one-of-controlled-ventilation)
    (weaning)
    (One-of-cpap-extubation))}

(PLAN one-of-controlled-ventilation ...
  ...
  (DO-SOME-ANY-ORDER
    (controlled-ventilation)
    (permissive-hypercapnia)
    (crisis-management)
  CONTINUATION-CONDITION controlled-ventilation)}

(PLAN controlled-ventilation
  (PREFERENCES (SELECT-METHOD BEST-FIT))
  (INTENTION:INTERMEDIATE-STATE (MAINTAIN STATE(BG) NORMAL controlled-ventilation *))
  (INTENTION:INTERMEDIATE-ACTION (MAINTAIN STATE(RESPIRATOR-SETTING) LOW controlled-ventilation *))
  (SETUP-PRECONDITIONS (PIP (<= 30) I-RDS *now*)
    (BG available I-RDS [[_, _], [_, _], [1 MIN,_] (ACTIVATED initial-phase-1#)]))
  (ACTIVATED-CONDITIONS AUTOMATIC)
  (ABORT-CONDITIONS ACTIVATED
    (OR (PIP (> 30) controlled-ventilation [[_, _], [_, _], [30 SEC, _], *self*])
       (RATE(BG) TOO-STEEP controlled-ventilation [[_, _], [_, _], [30 SEC,_], *self*])))
  (SAMPLING-FREQUENCY 10 SEC))
  (COMPLETE-CONDITIONS
    (FiO2 (<= 50) controlled-ventilation [[_, _], [_, _], [180 MIN, _], *self*])
    (PIP (<= 23) controlled-ventilation [[_, _], [_, _], [180 MIN, _], *self*])
    (f (<= 60) controlled-ventilation [[_, _], [_, _], [180 MIN, _], *self*])
    (state(patient) (NOT DYSPNEIC) controlled-ventilation [[_, _], [_, _], [180 MIN, _], *self*]))
    (STATE(BG) (OR NORMAL ABOVE-NORMAL) controlled-ventilation [[_, _], [_, _], [180 MIN,_], *self*])
    (SAMPLING-FREQUENCY 10 MIN))
  (DO-ALL-SEQUENTIALLY
    (one-of-increase-decrease-ventilation)
    (Observing)))
```

Figure 10 A portion of the I-RDS guideline, encoded in Asbru [18].

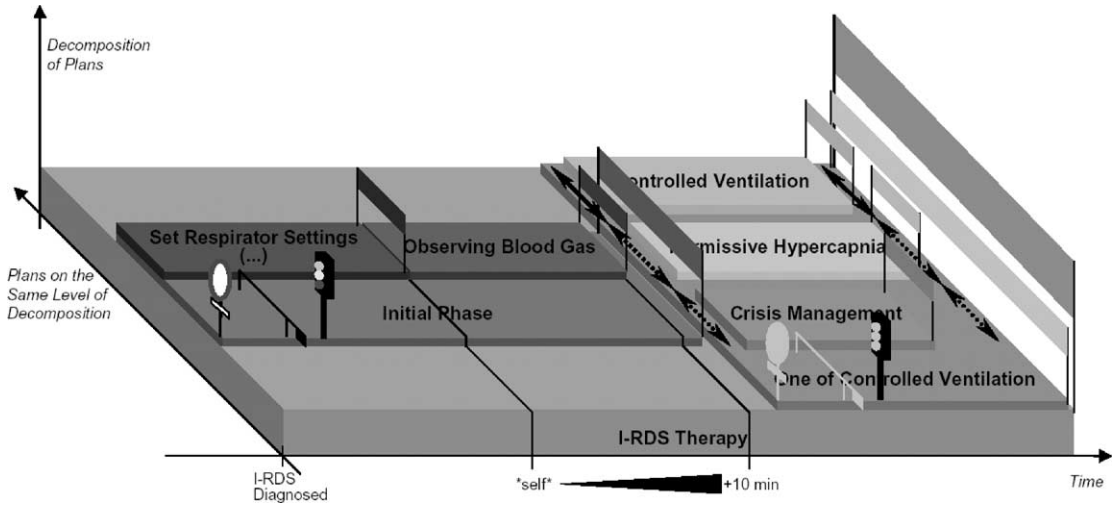

Figure 11 The I-RDS guideline, visualized in AsbruView. Of the four I-RDS subplans, only the 'initial-phase' and the 'one-of-controlled-ventilation' plans are shown here [66].

In AsbruView, plans can also be depicted in the Temporal view [67]. The Temporal view focuses on the temporal dimensions of plans and conditions by showing plans as explicit guidelines. The Temporal view uses the time annotation that specifies four points in time relative to a reference point. Fig. 12 shows a portion of the I-RDS guideline, visualized through the Temporal view.

As the Asbru language is formally defined, guidelines can be verified to detect various types of logical and procedural errors [68]. These tools are currently under development.

6.4. Guideline execution

Tools and software that facilitate the execution of guidelines, written in Asbru are currently under development.

7. EON

7.1. Introduction

7.1.1.Overview

EON, also developed at Stanford University, is a component-based architecture used to build decision support systems that reason about guideline-directed care [19]. The EON architecture consists of several components that facilitate the acquisition and execution of clinical guidelines. Similar to GLIF (as mentioned earlier, EON was one of the approaches from which GLIF originated), the guideline model of EON, called Dharma [21], is object-oriented and consists of classes that describe guideline entities as a sequence of structured temporal steps. The Dharma model is non-monolithic, meaning that the guideline model can be extended with

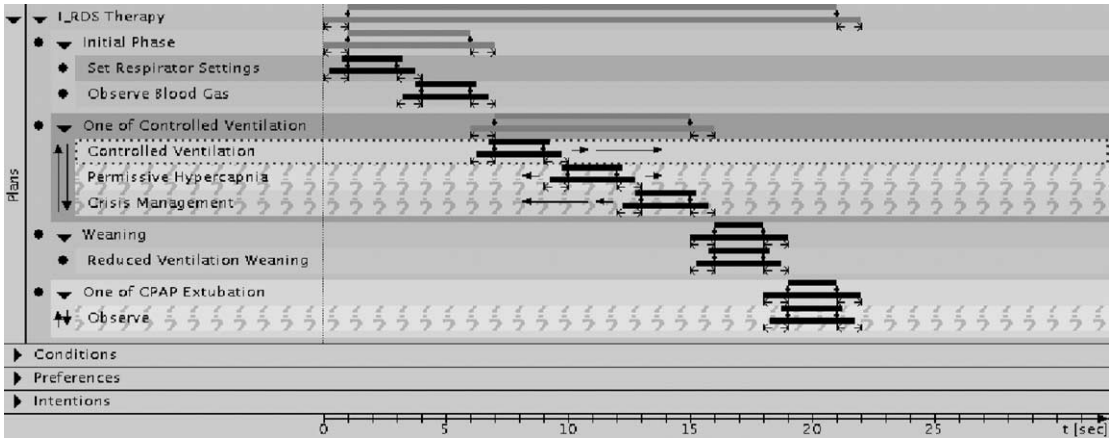

Figure 12 The I-RDS guideline, visualized in AsbruView through the temporal view [67].

additional classes that capture new guideline behavior. Besides the Dharma guideline model, the EON architecture also contains a number of run-time components, used to construct execution-time systems.

7.2. Guideline model and representation

7.2.1.The Dharma guideline model

In contrast with for example GLIF and PRO*forma* that model guidelines in terms of a fixed number of primitives (e.g., decisions, actions), the researchers of EON propose a non-monolithic (non-closed) guideline model, which consists of a standard set of primitives that can be extended with task-specific submodels, resulting in additional classes of primitives that are matched to the knowledge requirements of different guidelines.

In the Dharma model, guidelines manage patient behavior, consisting of decisions and actions that may lead to dependent changes in patient states over time (Fig. 13).

In this conceptual model of multi-encounter patient management, decisions are made during encounters between healthcare providers and patients. Actions such as writing a prescription or requesting a laboratory test, are carried out during encounters and may (in)directly lead to a change in the patient state (e.g., the progression of a disease). Some actions can start activities that extend over time. In order to define guidelines according to this conceptual model, they are represented in terms of a number of key characteristics, represented by primitives. Examples of EON primitives are scenarios, decisions, actions and goals. These primitives form the core guideline ontology.

7.2.1.1. Scenarios

A scenario is a (partial) characterization of the state of a patient (e.g., the patient is currently being prescribed a low-dosed steroid). In a scenario, eligibility conditions specify the necessary conditions for a patient to be in this scenario. Scenarios allow a clinician to synchronize the management of a patient with the corresponding parts of (a portion of) a guideline and are commonly used as entry points in a guideline. In the Dharma ontology, a scenario is always followed by a decision or action step. Each scenario in an actual guideline is an instance of the *scenario* class, which contains several attributes such as an attribute that specifies the eligibility criteria and an attribute that specifies the step that follows the current scenario (similar to GLIF). Scenarios allow a clinician to synchronize the management of a patient to situations handled by a guideline.

Scenarios can also serve to model exceptions, which represent exceptional situations that rarely occur. As expressing everything in a guideline can be impractical, a guideline author may want to partition the guideline into normal situations that cover usual cases and exceptions. The Dharma ontology defines two classes of exceptions: (1) exceptions that are repairable (i.e., those that lead a patient back to a scenario covered by the guideline), and (2) exceptions that are not repairable, so that patient has to be managed outside this guideline.

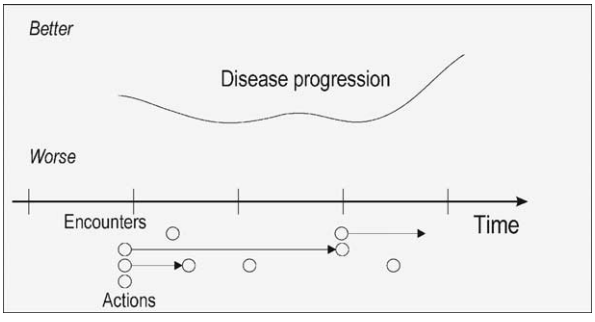

Figure 13 Conceptual model of the patient management process.

7.2.1.2. Decisions

In the Dharma core ontology, two basic types of decisions are defined (by means of two subclasses): decisions that model '*if-then-else*' choices and decisions that require making a heuristic choice from a set of pre-enumerated alternatives. The latter is aided by preferences as determined by rule-in and rule-out conditions that support or oppose each alternative (similar to the concept of schemas in PRO*forma*). If a rule-out condition evaluates to true, then the corresponding alternative is rejected. If the rule-out condition does not apply and a rule-in condition evaluates to true, the corresponding alternative is marked as preferred. If neither evaluates to true, then the preference for the choice can be determined by a default preference associated with each alternative.

7.2.1.3. Actions

Actions are instantaneous acts that lead to changes in the state of the world such as collecting patient data, displaying a message to the user or starting a drug regimen. Actions are used heavily throughout guidelines modeled in EON.

Whereas actions refer to instantaneous acts, activities model processes that take place over time. Activities have states that can change from time to time. These changes are usually the result of actions specified in a guideline, as actions are able

to start a new activity, stop an ongoing activity or change the attribute values of an ongoing activity.

Finally, the model also includes actions that refer to a set of other actions or a subguideline. Similar to GLIF, examples of such actions are actions that model branching and synchronization constructs in order to execute parallel tasks.

7.2.1.4. Goals

Every step can be associated with a goal. The notion of goals is comparable with the notions of intentions in Asbru, although less sophisticated. In the Dharma ontology, goals are represented as boolean criteria (e.g., '*reduce the arterial blood pressure to less than 130/85 within 3 weeks*'). The format of these criteria is explained later.

7.2.2. The patient data model

The patient data model defines classes and attributes in order to represent patient data. For example, the patient data model defines a *Patient* class, whose instances hold demographic information about specific patients, a *Qualitative_Entry* class that describes qualitative observations about patients, a *Numeric_Entry* class that stores results of quantitative measurements, an *Adverse_Event* class that models adverse reactions to specific substances, a *Condition* class that represents medical conditions that persist over time, and two intervention classes, *Medication* and *Procedure*, that model drugs and other medical procedures that have been recommended, or used. The patient data model defines characteristics regarding demographic and clinical conditions of specific patients. It does not aim at modeling the entire patient (e.g., replicate the structure of an EPR), but models only those distinctions that are relevant for the purpose of defining guidelines and protocols.

7.2.3. Medical-specialty model

The medical-specialty model consists of a medical domain ontology that models the structure of domain concepts (e.g., drugs and treatments) in terms of organized classes, relations and attributes. The medical-specialty model represents different sorts of domain-specific information.

7.2.4. Modeling tools

Protégé is used as a modeling tool to define the classes and attributes that form the core guideline model, the patient data model and the medical-specialty model. Also, additional classes that are derived from the core guideline model that introduce additional functionality are defined and entered by means of Protégé.

7.2.5. Guideline representation

Similar to GLIF, guidelines are represented in EON by temporally sequenced graphs (flowcharts) of instantiated classes. Regarding temporal aspects, EON has adopted a subset of the Asbru temporal language to represent temporal information.

7.2.6. Language

The EON model uses the internal frame-based Resource Description Format (RDF) of Protégé [69] to describe the models as well as the guidelines. Although the focus of the EON project is not on defining a formal syntax for representing guidelines in general, it does particularly address the subject of how to describe criteria that are used in decisions. EON defines three different criterion languages.

First, common but relatively simple criteria can be expressed as boolean criteria in terms of a set of object templates. Criteria encoded in this object-based language evaluate to true, false, or unknown. An example of such a criterion is '*diabetes mellitus is present and the most recent serum creatinine is less than normal*'.

According to the researchers of the EON project, such a criterion language is not expressive enough to capture more complex criteria such as '*is an authorized medication present that is contraindicated by some medical condition*'. To represent such criteria, the Protégé Axiom Language (PAL) is used, which is embedded into the Protégé development environment. The PAL constraint language is a subset of the Knowledge Interchange Format (KIF) syntax [70].

Finally, it is possible to write complex temporal criteria such as '*presence of an episode of uncontrolled blood pressure that overlaps with lisinopril medication and that started within 2 weeks after the initiation of lisinopril*'. These are written as temporal queries, which during guideline execution are translated to database queries [71].

Examples of criteria that were written in PAL or as temporal queries are shown in the next two sections.

7.3. Guideline acquisition, verification and testing

Besides defining the various EON models, Protégé is also used as a Knowledge Acquisition tool where guideline authors are able to enter and view guidelines. Protégé takes as input the Dharma guideline model, a Patient Data Model and a Medical-Specialty Model to create a Knowledge Acquisition Tool. Fig. 14 shows portion of a guideline for the treatment of breast-cancer, visualized in Protégé (see also Fig. 5).

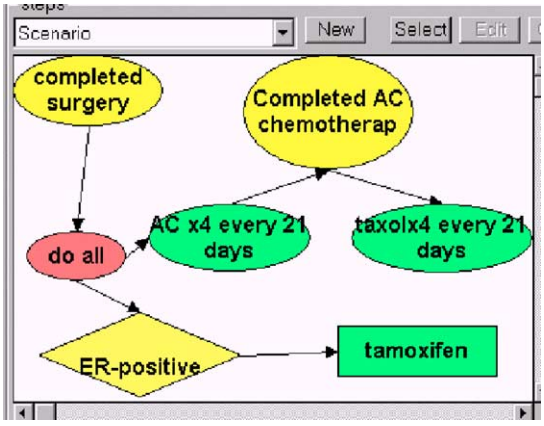

Figure 14 Part of a guideline that addresses the management of breast-cancer.

In this figure, the oval entitled '*do all*' represents a specific action step that models branching. Furthermore, statements such as '*taxol x4 every 21 days*' are repetitions of actions involving the drug taxol as a prescribable item. Besides actions, the part of the breast-cancer management protocol, shown in Fig. 14 also contains 2 scenarios ('*completed surgery*' and '*completed AC chemotherap*') and a decision ('*ER-positive*').

As mentioned in the previous section, PAL is used to formally describe complex criteria, used in decisions. PAL makes full use of Protégé's frame-based knowledge model. For example, variables can range over instances of Protégé classes and attributes of classes. Protégé contains a structured editor that facilitates guideline developers in writing these complex logical criteria. However, such criteria are usually not formulated and entered by domain experts that are not trained in logic.

7.4. Guideline execution

To facilitate the development of guideline execution engines, EON defines an execution architecture that contains components for guideline execution and interfacing to third-party information systems. Fig. 15 shows an overview of the execution architecture [72].

The heart of the execution architecture is formed by the Padda Guideline Execution Server (or Padda Server), which applies a clinical guideline to patient data queried from an information system's database and generates advisories [73]. Within the Padda Server, a knowledge-base handler manages access to the guideline knowledge base and the patient data model via the application-programming interface provided by Protégé. For a specific guideline and patient, the Padda Server must determine if the guideline is applicable to the patient, and subsequently, implement a model of interaction with the outside world (e.g., information systems or clinicians). The Padda Server uses patient data to suggest that a patient is in a specific scenario, and that, as a result, tasks such as laboratory tests should be performed. Users are always allowed to override the system's conclusions. For the Padda server to communicate with other information systems (e.g., EPRs), an interface specification has been defined. This specification, written in Common Object Request Broker Architecture Interface Definition Language (CORBA IDL) consists of methods with which client and server interact with each other as well as a description of the data structures that are passed between the server and clients.

To evaluate specific patient situations, available patient data must be mapped to the terms and

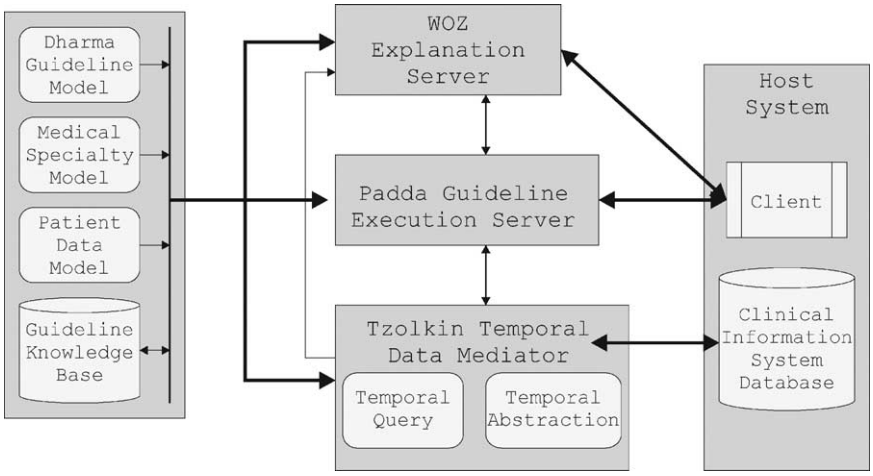

Figure 15 An overview of the EON execution architecture.

relations that are used in the guideline. The linking of concepts in the patient data model to corresponding concepts in an information system (e.g., an EPR) is done through the Tzolkin data mediator [71]. This component performs two functions. First, it maps concepts from a particular patient data model to corresponding concepts from the data model of the host system. Second, it maps terminology in the medical-specialty model (e.g., as names of laboratory test results) to the terminology used in the host information system. As mentioned earlier, criteria often contain complex temporal expressions. For formulating these types of criteria, the Tzolkin temporal data mediator contains a temporal query language that is able to define such temporal expressions. During runtime, the Tzolkin data mediator translates these temporal queries to 'standard' database queries (e.g., SQL). Finally, the WOZ (Wizard of OZ) component provides explanation services [74].

8. Discussion

8.1. Comparison

8.1.1.Overview

Each approach focuses on different aspects of guideline representation, development and implementation. The Arden Syntax and GLIF approaches focus on guideline standardization, PRO*forma* on execution aspects, Asbru on the representation and visualization of complex temporal plans, and EON on the development of an architecture that supports the development and implementation of guidelines. These different focus points have their implications regarding the representation, acquisition, verification and implementation of guidelines as shown in the previous sections.

8.1.2. Guideline modeling and representation

8.1.2.1. Primitives

The representation model of the Arden Syntax differs from other approaches, as it is the only approach that models each guideline as an independent modular rule. As a result, the Arden Syntax is most suitable for representing simple guidelines such as alerts in reminder systems.

The GLIF, PRO*forma*, Asbru and EON approach all model guidelines in a similar way, in terms of primitives (steps, tasks or plans) that describe the control structure of a guideline. GLIF and EON have very similar models, as they were partly developed by the same groups and researchers. The main difference is that the GLIF model, just as PRO*forma* and Asbru, contains a fixed number of primitives, while the EON set of primitives is extensible. The basic primitives however such as primitives that represent decisions, actions and patient states (entry points) are present in both GLIF and EON. MLMs contain similar constructs such as decisions (*logic* slot), actions (*action* slot) and patient states (*evoke* slot). Primitives that describe decisions and actions are also present in PRO*forma*. Although PRO*forma* does not provide explicit support for defining patient states, it is possible to model these through constructs like triggers and pre- and postconditions [75]. The PRO*forma* enquiry task is viewed as an action in the GLIF and EON models. In Asbru, the basic primitive is an action: every (sub)plan eventually consists of actions. In contrast to other approaches where the functionality is described in terms of primitives, Asbru uses knowledge roles such as preferences, intentions, conditions and effects for this purpose.

All approaches support some form of temporal reasoning, of which the Asbru approach contains the most sophisticated structures. EON and GLIF both adopt a subset of the Asbru temporal language. In order to be compatible with the Arden Syntax, the GLIF expression language (GEL) also defines a number of operators that are defined in the Arden Syntax such as '*before*', '*after*' and '*ago*'. Similar constructs are also available in the PRO*forma* expression language. The Arden Syntax and GLIF support a limited form of uncertainty in terms of a three-valued logic ('*true*', '*false*' and '*unknown*'). PRO*forma* is the only approach that contains expressive constructs for describing uncertainty aspects of a guideline. In contrast with the issue of representing temporal aspects, the representation of uncertainty in guidelines is not regarded as a critical issue in general.

8.1.2.2. Complexity

All models except for the Arden Syntax provide explicit support for nesting of guidelines in order to model complex guidelines in terms of subguidelines (GLIF and EON) or subplans (PRO*forma* and Asbru). For this purpose, GLIF, EON and PRO*forma* contain an *Action* primitive that may contain a reference to a subguideline or subplan. In Asbru, each plan body contains a number of subplans until a non-decomposable plan (also called *Action*) is encountered. Although the Arden Syntax is able to call other rules in the *Action* slot, there is no general way of controlling these invocations.

EON, PRO*forma* and Asbru also support the use of goals and intentions to formally specify a guideline on a higher level of abstraction. Of these

techniques, the Asbru intension model is the most sophisticated.

GLIF defines different layers of abstraction, which allows guideline authors to view only the general control structure (flowchart) of a guideline before specifying all the necessary details. EON uses a non-monolithic approach: the Dharma guideline model is based on a core model, which can be extended with submodels depending on the complexity of the guideline (e.g., '*if-then-else*' rules versus complex treatment guidelines).

The representations of Asbru and EON also allow for the abstraction of temporal data to facilitate the specification of complex temporal expressions.

Except Asbru, all approaches support the concept of referenced subguidelines. In Asbru, subplans are 'embedded' in a plan, meaning that this subplan is not known outside the embedding plan. As a result, a certain subplan is not sharable with other plans outside the embedding plan.

GLIF also supports the representation of common guideline structures through Macros, which facilitates the reuse of guidelines that are used often (e.g., '*if-then*' rules such as MLMs).

8.1.2.3. Knowledge types

Besides the knowledge that defines the control structure (e.g., rules, primitives, plans, sequences), every guideline also contains domain-specific knowledge such as medical knowledge (e.g., terminology) and knowledge concerning the patient (e.g., the patient's symptoms or history).

In the Arden Syntax each reference to a domain-specific item is stored as a label in the *data* slot of a MLM. As a result, a MLM does not 'know' for example that amoxicillin is an antibiotic. Also PRO*forma* and Asbru contain no explicit support for modeling domain-specific knowledge or for using standard terminology systems. GLIF addresses this problem by modeling domain-specific knowledge by means of defining a Medical Ontology that contains three different layers: the core GLIF layer, the RIM layer and the medical knowledge layer. EON takes a very similar approach by defining the Dharma guideline model, the Patient Data Model and the Medical-Specialty Model. Currently, both the layers in GLIF as well as the models in EON are still partly under development.

Besides invoking subguidelines, a guideline may consist of various types of actions such as medically oriented actions (e.g., recommending a particular course of treatment) and programming-oriented actions (e.g., supplying a message to a care provider). In the Arden Syntax, actions (stored in the *action* slot) are usually programming-oriented as they are used to generate reminders or alerts. This is also the case in the PRO*forma* approach, as a PRO*forma* action is a programming-related task that is carried out by the execution engine through an Application Programming Interface (API). GLIF and EON both support these two types of actions. Finally, Asbru does not support programming-related actions.

8.1.2.4. Didactic and maintenance

Didactic and maintenance information concerns information about authors, versioning, purposes and detailed explanations. The Arden Syntax, GLIF and EON approaches are all able to hold various kinds of information such as the guideline's author, version, institution, keywords, validation (e.g., '*research*', '*testing*', '*production*') and explanation. In PRO*forma* and Asbru, it is not possible to store didactic and maintenance-related information (besides a name and explanation).

8.1.2.5. Language

All approaches except EON have defined a language that entirely describes the representation through a formal syntax: the Arden Syntax, PRO*forma* and Asbru use BNF (the latest version of Asbru is also in XML-format) and GLIF uses UML. EON relies on the internal syntax of Protégé. For each approach, the syntax captures all aspects that are defined in the corresponding representations.

Regarding the guidelines itself, the Arden Syntax describes guidelines in terms of a semi-structured ASCII format (see Fig. 1), GLIF describes guidelines in an XML format (see Fig. 4), Asbru in a LISP-like syntax (see Fig. 10) and PRO*forma* in the R^2L language. EON uses a description that is very similar to that of GLIF, with the main exception that GLIF describes expressions in the guideline expression language (GEL) while EON describes expressions by means of the three different criterion languages.

PRO*forma* is the only approach, which makes a distinction between a declarative language (e.g., R^2L), used during the guideline acquisition phase and a procedural language (e.g., L_{R2L}) that is processed by a general interpreter (e.g., PROLOG) in an execution engine. All other approaches require a custom-developed execution engine, in which the different procedural aspects of the guideline are encoded programmatically (e.g., a number of Java or C procedures that each executes a certain primitive).

In order to facilitate the translation from a declarative language to a procedural language, the PRO*forma* representation language contains constructs that are filled in during guideline acquisition but are execution-related. For example, PRO*forma* defines an execution state that denotes the

state of a guideline during execution (e.g., '*in progress*', '*aborted*', '*terminated*', '*performed*'). This is in contrast with EON and GLIF that define patient states which are used during execution to determine the applicability of a guideline (as mentioned earlier, PRO*forma* is also able to model patient states implicitly through constructs like triggers and pre- and postconditions). Similar to PRO*forma*, Asbru also contains the concept of guideline execution states.

8.1.3. Guideline acquisition

The developers of the Arden Syntax have not developed tools that facilitate the process of guideline acquisition, although various acquisition tools were created by third parties. GLIF and EON use Protégé as the main knowledge acquisition tool (see Figs. 5 and 14), in which guidelines are entered as flowcharts. As mentioned earlier, every primitive is shown as a generic form in Protégé. The advantage is that the user interface is created automatically by Protégé. The disadvantage is that there is limited guidance as each instance is shown as a separate form and guideline authors can get 'lost' when there are too many forms open.

The GLIF expression language is similar to the Arden Syntax. During knowledge acquisition in Protégé, GLIF expressions are still entered as strings (e.g., '*test_name* ='*Serum_Potassium*'''), which has to be parsed in order to extract the various kinds of information such as the different operators and used domain terms. Therefore, it is possible for guideline authors to type in erroneous criteria if there is no syntax checker available. As EON contains three different criterion languages, each guideline author has to decide which of these three languages (s)he will use. Although the EON architecture contains tools that partly facilitate the structured entry of these languages, only guideline authors that are skilled in writing logic and database queries will be able to write complex criteria using PAL logic or temporal queries.

The PRO*forma* tasks are very basic and 'low-level', so that it may be difficult for guideline authors to enter guidelines, as they often do not view guidelines in terms of schemas, pre-and post conditions and predicate logic, making PRO*forma* more like a guideline programming language than an abstract representation. PRO*forma* contains a very elaborate tool for guideline acquisition. The acquisition tool facilitates guideline authors using a sophisticated graphical editor, as shown in Fig. 7. However, guideline authors may interpret the constraint satisfaction graph as a standard flowchart. This is not the case however, as the arrows between task instances can represent different types of constraints. Also, guideline authors are required to specify execution-time information such as guideline execution states, which may differ from an author's viewpoint of a guideline.

Acquiring guidelines by means of using graphical metaphors has become one of the focus points of Asbru. AsbruView uses sophisticated visualization techniques to facilitate the acquisition of complex guidelines. In contrast to GLIF, EON and PRO*forma*, the Asbru researchers have chosen not to model guidelines through flowcharts but by means of metaphor graphics such as running tracks or traffic lights. It has still to be proven which visualization technique is the most suited.

8.1.4. Guideline verification and testing

PRO*forma* is the only approach that has developed tools, which—based on a sound formal language—verify entered guidelines by detecting a number of possible logical and procedural errors such as incorrect data types, invalid syntax or attribute values, critical missing values or concepts and inconsistent constraints. Although for other approaches, tools for guideline verification and testing are reported to be in development, no results have been published so far.

8.1.5. Guideline execution

EON and PRO*forma* have developed execution engines which are able to process guidelines developed in the corresponding languages. Also, these two approaches have published results on the development and implementation of actual decision support systems. PRO*forma* is the only approach that has developed a commercialized version. Both systems are able to communicate with clinical information systems and users through standard Application Programming Interfaces (APIs) or communication protocols (e.g., CORBA).

A number of third parties have implemented decision support systems that are able to execute Arden Syntax guidelines for use in their local institutions. However, these are often not reusable in other environments.

Sharing of guidelines is possible when all guidelines are encoded in the same guideline representation language. Since no existing guideline representation model is dominant over the others, Wang et al. [76] proposed an alternative approach, the Guideline Execution by Semantic Decomposition of Representation (GESDOR) model, to guideline sharing at the execution level. This approach is based on the observation that the different guideline representation models contain similar execution tasks. A set of generalized execution tasks are extracted from the existing guideline representa-

tion models. This set is then used to drive the execution of specific guidelines encoded in different formats. The effectiveness of the approach was confirmed using the GLIF3 model and the PRO*forma* model as the two prototype guideline representation models. Another approach suggested by Fox et al. is to publish computer-interpretable guidelines as web-accessible services, called publets [77].

As mentioned earlier, no publications are known that address the development of guideline execution engines, which are able to execute Asbru guidelines.

8.2. Conclusions

In the last decade, most of the attention is focused on the areas of guideline representation models and underlying languages. However, the real benefit lays in structuring and guiding the whole guideline development process: in order to successively implement decision support systems that will be used in daily practice, all the four areas (representation, acquisition, verification and execution) must be taken into account. This is not a trivial task. Comparing the various approaches, mentioned in this paper shows that design specifications made in one area (e.g., guideline representation) have implications in other areas (e.g., guideline execution).

For example, Asbru defines a guideline representation language that has a very rich set of temporal constructs. However, a general guideline execution engine still has to be developed that can be used in daily practice. Another example is PRO*forma* that focuses on guideline execution. This is reflected in the guideline model: each primitive in the PRO*forma* task ontology can easily be mapped to a corresponding component in a guideline execution engine. However, during guideline acquisition, all guidelines have to be defined in terms of those primitives, which makes PRO*forma* a more low-level language.

Although significant progress has been made during the last years, especially regarding guideline representation, several issues that relate to guideline implementation and guideline-based decision support still have to be addressed more extensively. Examples of such issues are how to implement national guidelines as well as local adaptations of those guidelines and how to increase the shareability of generic guideline execution engines among different intuitions. Various solutions may be developed that address these issues such as the development of versioning methods that enable synchronization between national and local guidelines and the development of standard interfaces to different external information systems.

In order to create an approach that is successful, an acceptable compromise between all areas must be reached with the above-mentioned aspects as starting points. In this compromise, a balance must be maintained between the aspects of abstractness, expressiveness, formalization, acquisition and execution.

References

[1] Grimshaw JM, Russel IT. Effects of clinical guidelines on medical practice: a systematic review of rigorous evaluation. Lancet 1993;342:1317—22.
[2] Grimshaw JM, Russel IT. Implementing clinical practice guidelines: can guidelines be used to improve clinical practice? Effective Health Care 1994;8:1—12.
[3] Audet A, Greenfield S, Field M. Medical practice guidelines: current activities and future directions. Ann Intern Med 1990;113:709—14.
[4] Vissers MC, Hasman A, van der Linden CJ. Impact of a protocol processing system (ProtoVIEW) on clinical behaviour of residents and treatment. Int J Biomed Comput 1996;42(1—2):143—50.
[5] East TD, Henderson S, Pace NL, Morris AH, Brunner JX. Knowledge engineering using retrospective review of data: a useful technique or merely data dredging? Int J Clin Monit Comput 1991;8(4):259—62.
[6] Field MJ, Lohr KN, editors. Guidelines for clinical practice: from development to use. Washington, DC: National Academy Press; 1992.
[7] Van der Lei J, Talmon JL. Clinical decision-support systems. In: Van Bemmel JH, Musen MA, editors. Handbook of medical informatics. Houten: Bohn Stafleu Van Loghum; 1997.
[8] Fridsma DB, Gennari JH, Musen MA. Making generic guidelines site-specific. In: Proceedings of the AMIA Symposium; 1996. p. 597—601.
[9] Position statements from the Invitational Workshop: towards representations for sharable guidelines. Available at http://www.glif.org/workshop/statement.htm.
[10] Wang D, Peleg M, Tu SW, Boxwala AA, Greenes RA, Patel VL, Shortliffe EH. Representation primitives process models and patient data in computer-interpretable clinical practice guidelines: a literature review of guideline representation models. Int J Med Inf 2002;68(1—3):59—70.
[11] Peleg M, Tu S, Bury J, Ciccarese P, Fox J, Greenes RA, Hall R, Johnson PD, Jones N, Kumar A, Miksch S, Quaglini S, Seyfang A, Shortliffe EH, Stefanelli M. Comparing computer-interpretable guideline models: a case-study approach. J Am Med Inform Assoc 2003;10(1):52—68.
[12] Purves IN, Sugden B, Booth N, Sowerby M. The PRODIGY project—the iterative development of the release one model. In: Proceedings of the AMIA Symposium; 1999. p. 359-63.
[13] Quaglini S, Stefanelli M, Cavallini A, Micieli G, Fassino C, Mossa C. Guideline-based careflow systems. Artif Intell Med 2000;20(1):5—22.
[14] Herbert SI, Gordon CJ, Jackson-Smale A, Salis JL. Protocols for clinical care. Comput Methods Programs Biomed 1995;48(1—2):21—6.
[15] Clayton PD, Pryor TA, Wigertz OB, Hripcsak G. Issues and structures for sharing knowledge among decision-making

systems: The 1989 Arden Homestead Retreat. In: Kingsland LC, editor. Proceedings of the Thirteenth Annual Symposium on Computer Applications in Medical Care. New York: IEEE Computer Society Press; 1989. p. 116—21.
[16] Ohno-Machado L, Gennari JH, Murphy SN, Jain NL, Tu SW, Oliver DE, Pattison-Gordon E, Greenes RA, Shortliffe EH, Barnett GO. The guideline interchange format: a model for representing guidelines. JAMIA 1998;5(4):357—72.
[17] Fox J, Johns N, Rahmanzadeh A. Disseminating medical knowledge: the PRO*forma* approach. Artif Intell Med 1998; 14:157—81.
[18] Shahar Y, Miksch S, Johnson P. The Asgaard Project: a task-specific framework for the application and critiquing of time-oriented clinical guidelines. Artif Intell Med 1998; 14:29—51.
[19] Musen MA, Tu SW, Das A, Shahar Y. EON: a component-based approach to automation of protocol-directed therapy. JAMIA 1996;3:367—88.
[20] De Clercq PA, Hasman A, Blom JA, Korsten HHM. Design and implementation of a framework to support the development of clinical guidelines. Int J Med Inform 2001;64:285—318.
[21] Tu SW, Musen MA. A flexible approach to guideline modeling. In: Proceedings of the AMIA Symp; 1999. p. 420—4.
[22] Boxwala AA, Tu SW, Zeng Q, Peleg M, Ogunyemi O, Greenes RA, Shortliffe EH, Patel VL. Towards a representation format for sharable clinical guidelines. J Biomed Inform 2001;34(3):157—69.
[23] De Clercq PA, Blom JA, Hasman A, Korsten HHM. A strategy for development of practice guidelines for the ICU using automated knowledge acquisition techniques. Int J Clin Monit Comput 1999;15:109—17.
[24] De Clercq PA, Hasman A. Experiences with the Development, Implementation and Evaluation of Automated Decision Support Systems. Medinfo; 2004, in press.
[25] Pryor TA, Gardner RM, Clayton PD, Warner HR. The HELP system. In: Blum BI, editor. Information systems for patient care. New York: Springer Verlag; 1984. p. 109—28.
[26] McDonald C, Overhage JM, Dexter PR, Tierney WM, Suico JG, Zafar A, Schadow G, Blevins L, Warvel J, Meeks-Johnson J, Lemmon L, Glazener T, Belsito A, Lindbergh D, Williams B, Cassidy P, Xu D, Tucker M, Edwards M, Wodniak C, Smith B, Hogan T. The Regenstrief Medical Record System 1999: sharing data between hospitals. In: Proceedings of the AMIA Symposium, vols. 1—2; 1999. p. 1212.
[27] Hripcsak G, Ludemann P, Pryor TA, Wigertz OB, Clayton PD. Rationale for the Arden Syntax. Comput Biomed Res 1994;27(4):291—324.
[28] Peleg M, Boxwala AA, Bernstam E, Tu S, Greenes RA, Shortliffe EH. Sharable representation of clinical guidelines in GLIF: relationship to the Arden Syntax. J Biomed Inform 2001;34(3):170—81.
[29] Clinical Decision Support & Arden Syntax Technical Committee of HL7, inventor Arden Syntax for Medical Logic Systems, version 2.0. Draft revision. USA: July 7; 1999.
[30] Pryor TA, Hripcsak G. Sharing MLMs: an experiment between Columbia-Presbyterian and LDS Hospital. In: Proceedings of the Annual Symposium on Computer and Applied Medical Care; 1993. p. 399—403.
[31] Sherman EH, Hripcsak G, Starren J, Jenders RA, Clayton P. Using intermediate states to improve the ability of the Arden Syntax to implement care plans and reuse knowledge. In: Proceedings of the Annual Symposium on Computer and Applied Medical Care; 1995. p. 238—42.
[32] Gao X, Shahsavar N, Arkad K, Ahlfeldt H, Hripc sak GWigertz O. Design and function of medical knowledge editors for the Arden syntax. Medinfo 1992:472—7.
[33] Jenders RA, Dasgupta B. Assessment of a knowledge-acquisition tool for writing Medical Logic Modules in the Arden Syntax. In: Proceedings of the AMIA Symposium; 1996. p. 567—71.
[34] Bang M, Eriksson H. Generation of development environments for the Arden Syntax. In: Proceedings of the AMIA Symposium; 1997. p. 313—7.
[35] Hripcsak G, Cimino JJ, Johnson SB, Clayton PD. The Columbia-Presbyterian Medical Center decision-support system as a model for implementing the Arden Syntax. Proceedings of the Annual Symposium on Computer and Applied Medical Care; 1991. p. 248—52.
[36] Kuhn RA, Reider RS. A C++ framework for developing Medical Logic Modules and an Arden Syntax compiler. Comput Biol Med 1994;24(5):365—70.
[37] Hripcsak G, Clayton PD. User comments on a clinical event monitor. Proceedings of the Annual Symposium on Computer and Applied Medical Care; 1994. p. 636—40.
[38] The Intermed Collaboratory. Homepage available at http://smi-web.stanford.edu/projects/intermed-web/.
[39] Peleg M, Boxwala AA, Ogunyemi O, Zeng Q, Tu S, Lacson R, Bernstam E, Ash N, Mork P, Ohno-Machado L. Shortliffe EH, Greenes RA, GLIF3: the evolution of a guideline representation format. In: Proceedings of the AMIA Symposium; 2000. p. 645—9.
[40] Stoufflet PE, Ohno-Machado L, Deibel SRA, Lee D, Greenes RA. GEODE-CM: A state-transition framework for clinical management. In: Cimino JJ, editor. Proceedings of the 20th Annual Symposium on Computer Applications in Medical Care. Philadelphia: Hanley and Belfus; 1996. p. 924.
[41] Barnes M, Barnett GO, An architecture for a distributed guideline server In: Miller RA, editor. Proceedings of the 19th Annual Symposium on Computer Applications in Medical Care. Philadelphia: Hanley and Belfus; 1995. p. 233—7.
[42] Object Management Group. Unified Modeling Language (UML) specification. Available at http://www.rational.-com/uml/index.jtmpl. version 1.3 1999.
[43] Peleg M, Ogunyemi O, Tu SW, Boxwala AA, Zeng Q, Greenes RA, Shortliffe EH. Using features of arden syntax with object-oriented medical data models for guideline modeling. In: Proceedings of the AMIA Symposium; 2001. p. 523—7.
[44] Schadow G, Russler DC, Mead CN, McDonald CJ. Integrating Medical Information and Knowledge in the HL7 RIM. In: Proceedings of the AMIA Symposium; 2000. p. 764—8.
[45] Lindberg C. The Unified Medical Language System (UMLS) of the National Library of Medicine. J Am Med Rec Assoc 1990;61(5):40—2.
[46] W3C. Extensible Markup Language (XML). Available at http://www.w3.org/XML/ 2000.
[47] Sordo M, Ogunyemi O, Boxwala AA, Greenes RA, GELLO: an object-oriented query and expression language for clinical decision support. In: Proceedings of the AMIA Symposium; 2003. p. 1012.
[48] Advisory Committee on Immunization Practices A.Prevention and Control of Influenza. Morbidity and Mortality Weekly Report 49(RR03);2000:1—38.
[49] Irwin RS, Boulet LS, Cloutier MM, Gold PM, Ing AJ, O'byrne P et al. Managing cough as a defense mechanism and as a symptom, a consensus panel report of the american college of chest physicians. Chest 1998;114(2):133S—81S.
[50] American College of Cardiology/American Heart Association/American College of Physicians/American Society of Internal Medicine. Guidelines for the management of patients with chronic stable angina. J Am Col Cardiol 1999;33:2092—2197.

[51] Grosso WE, Eriksson H, Fergerson RW, Gennari JH, Tu SW, Musen MA. Knowledge modeling at the millennium (the design and evolution of Protégé-2000). In: Proceedings of the 12th International Workshop on Knowledge Acquisition, Modeling and Mangement (KAW'99). Banff, Canada, October 1999.
[52] Greenes RA, Boxwala A, Sloan WN, Ohno-Machado L. Deibel SRAA. Framework and tools for authoring, editing, documenting, sharing, searching, navigating, and executing computer-based clinical guidelines. In: Proceedings of the AMIA Symposium; 1999. p. 261—5.
[53] Peleg M, Patel VL, Snow V, Tu S, Mottur-Pilson C, Shortliffe EH, Greenes RA. Support for guideline development through error classification and constraint checking. In: Proceedings of the AMIA Symposium; 2002. p. 607—11.
[54] Zielstorff RD, Teich JM, Paterno MD, Segal M, Kuperman GJ, Hiltz FL, Fox RL. P-CAPE: a high-level tool for entering and processing clinical practice guidelines. In: Proceedings of the AMIA Symposium; 1998. p. 478—82.
[55] Wang D, Shortliffe EH. GLEE—a model-driven execution system for computer-based implementation of clinical practice guidelines. In: Proceedings of the AMIA Symposium; 2002. p. 855—9.
[56] Fox J, Das S. Arguments about beliefs and actions: decision making in the real world. In: Fox J, Das S, editor. Safe and sound: artificial intelligence in hazardous applications; 2000. p. 55—76.
[57] Fox J, Das S. The RED knowledge representation language. In: Fox J, Das S, editors. Safe and sound: artificial intelligence in hazardous applications; 2000. p. 191—206.
[58] Fox J, Das S. Constructing intelligent systems. In: Fox J, Das S, editors. Safe and sound: artificial intelligence in hazardous applications; 2000. p. 77—116.
[59] Fox J, Das S. Safety first. In: Fox J, Das S, editors. Safe and sound: artificial intelligence in hazardous applications; 2000. p.191—206.
[60] Fox J, Das S. A formalization of safety. In: Fox J, Das S, editors. Safe and sound: artificial intelligence in hazardous applications; 2000. p. 77—116.
[61] Sutton DR, Fox J. The syntax and semantics of the PROforma guideline modeling language. JAMIA 2003; 10(5):433—43.
[62] InferMed. Homepage availabe at http://www.infermed.-com.
[63] Miksch S, Shahar Y, Johnson P. Asbru: a task-specific, intention-based, and time-oriented language for representing skeletal plans. In: Proceedings of the Seventh Workshop on Knowledge Engineering Methods and Languages (KEML-97). Milton Keynes, UK.
[64] Chandraskekaran B. Generic tasks in knowledge-based reasoning: high-level building blocks for expert system design. IEEE Expert 1986;1:23—30.
[65] Asbru 7.2 syntax. Available at http://www.ifs.tuwien.ac.at/asgaard/asbru/Syntax.
[66] Miksch S, Kosara R, Shahar Y, Johnson PD. AsbruView: visualization of time-oriented, skeletal plans. In: Proceedings of the Fourth International Conference on Artificial Intelligence Planning Systems. Pittsburgh, PA: Carnegie-Mellon University; 1998. p. 11—18.
[67] Kosara R, Miksch S. Metaphors of movement: a visualization and user interface for time-oriented, skeletal plans. Artif Intell Med 2001;22(2):111—31.
[68] Duftschmid G, Miksch S. Knowledge-based verification of clinical guidelines by detection of anomalies. Artif Intell Med 2001;22(1):23—41.
[69] World Wide Web Consortium, Resource Description Framework (RDF). Available at http://www.w3.org/RDF/.
[70] Knowledge Interchange Format: Draft Proposed American National Standard (dpANS); 1998.
[71] Nguyen JH, Shahar Y, Tu SW, Das AK, Musen MA. Integration of temporal reasoning and temporal-data maintenance into a reusable database mediator to answer abstract, time-oriented queries: the tzolkin system. J Intell Inform Sys 1999;13(1/2):121—45.
[72] Tu SW, Musen MA. Modeling data and knowledge in the EON Guideline Architecture. Medinfo 2001;10:280—4.
[73] Tu SW, Musen MA. From guideline modeling to guideline execution: defining guideline-based decision-support services. In: Proceedings of the AMIA Symposium; 2000. p. 863—7.
[74] Shankar RD, Musen MA. Justification of automated decision-making: medical explanation or medical argument? In: Proceedings of the AMIA Symposium; 1999. p. 395—9.
[75] Bury JP, Saha V, Fox J. Supporting 'scenarios' in the PROforma guideline modeling format. In: Proceedings of the AMIA Annual Symposium; 2001. p. 870—4.
[76] Wang D, Peleg M, Bu D, Cantor M, Landesberg G, Lunenfeld E, Tu SW, Kaiser GE, Hripcsak G, Patel VL, Shortliffe EH. GESDOR—a generic execution model for sharing of computer-interpretable clinical practice guidelines. In: Proceedings of the AMIA Annual Symposium; 2003. p. 694—8.
[77] Fox J, Bury J, Humber M, Rahmanzadeh A, Thomson R. Publets: clinical judgement on the web. In: Proceedings of the AMIA Annual Symposium; 2001. p. 179—83.