Published in final edited form as:
Stud Health Technol Inform. 2008 ; 139: 22–43.

Computer-interpretable Guideline Formalisms

Paul DE CLERCQ[a,1], **Katharina KAISER**[b], and **Arie HASMAN**[c]

[a]Medecs, the Netherlands [b]Institute of Software Technology and Interactive Systems, Vienna University of Technology, Austria [c]Department of Medical Informatics, University of Amsterdam, the Netherlands

Abstract

Implementing Computer-Interpretable Guidelines (CIGs) in active computer-based decision support systems promises to improve the acceptance and application of guidelines in daily practice. The model and underlying language are the core characteristics of every CIG approach. However, currently no standard model or language has been accepted by the CIG community. This aim of this chapter is to provide an overview of well-known approaches and to formulate a set of (minimal) requirements that can be used in the process of developing new CIG approaches or improving existing ones. It presents five CIG approaches (the Arden Syntax, GLIF, PRO*forma*, Asbru and EON), followed by a general discussion of the strong points of each approach as well as their implications for future research.

Keywords

Computer-interpretable Guidelines; Knowledge Representation; Decision Support Systems

Introduction

Computer-interpretable Guidelines

During the last decade, studies have shown the benefits of using clinical guidelines in the practice of medicine such as a reduction of practice variability and patient care costs, while improving patient care. A variety of guidelines have been developed that focus on different application domains as well as different modes of use.

Although the potential application of guidelines in daily care is enormous, a number of difficulties exist related to the development and implementation of guidelines. One of them is the interpretation of the content of a guideline: the exact meaning of terms is not always defined, recommendations are not always clearly articulated and sometimes vague wording is used. Most of these guidelines are written down as large documents in a textual format, which are often cumbersome to read and difficult to integrate and apply in the patient care process. Additional problems exist also, related to the areas of maintenance (e.g., updating and versioning) and (local) adaptation (e.g., adapting national guidelines to local protocols). Although the importance of guidelines is increasingly recognised, health care institutions often pay more attention to guideline development than to guideline implementation for routine use in daily care.

[1]Corresponding Author: Medecs, Horsten 2, 5612 AX Eindhoven, The Netherlands; p.a.d.clercq@medecs.nl.

Implementing guidelines in active computer-based decision support systems promises to improve the acceptance and application of guidelines in daily practice because these systems are able to monitor the actions and observations of care providers and to provide guideline-based advice at the point of care. It is stated that (guideline-based) decision support systems are in fact necessary for the future of medical decision making in general [1].

These so-called Computer-Interpretable Guidelines (CIGs) are increasingly applied in diverse areas and many parties are developing CIGs as well as decision support systems that incorporate these guidelines, covering a wide range of clinical settings and tasks (an overview can be found on OpenClinical [2]). Despite these efforts, only a few systems progressed beyond the prototype stage and the research laboratory. Building systems that are both effective in supporting clinicians and accepted by them has proven to be a difficult task.

Various questions arise when developing and implementing CIGs, such as:

- How to represent and share various types of guidelines using a formal and unambiguous representation;
- How to acquire, verify, localize, execute and evaluate formalised guidelines and support systems in daily practice;
- How to interface guideline-based decision support systems with external patient information systems;
- How to provide decision support to a care provider in daily practice.

Although these are all relevant and important questions, this chapter will focus mostly on the first questions, namely with respect to the issue of representing and sharing CIGs using a formal model. More information on other issues concerning the development and implementation of CIG decision support systems is described elsewhere [3].

CIG Approaches

Nowadays, many approaches exist for specifying CIGs, each with its own motivations and features [4]. For example, some approaches focus more on guideline standardisation and interoperability, while others focus more on guideline development or decision support. These different foci have their implications for the representation of CIGs.

This chapter will present and discuss a number of well-known CIG approaches, with the goal of providing a general comparison and discussion in order to identify the strong points of the various CIG approaches. Based on known approaches, reviews [3, 4, 5-8] and own experiences, it is possible to define the functionality of CIG approaches in terms of a two main characteristics: the underlying model and the language in which guidelines are specified.

The model is the core characteristic of every guideline approach. It 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. The model must contain a set of building blocks used to construct guidelines, such as tasks, rules, nodes or frames. For example, most approaches model guidelines in terms of a Task-Network Model (TNM): a (hierarchical) model of the guideline control flow as a network of specific tasks (e.g., flowchart) [6]. TNMs are typically based on a standard repertoire of generic tasks such as decisions and actions. The model must be expressive enough to represent these various aspects. Also, *g*uidelines 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). The model must support these types of knowledge and should model them separately to support guideline sharing and to ensure that guidelines can be used in multiple clinical domains and in various modes (e.g., proactive vs. reactive use) [9]. The model should also support aspects related to didactics and maintenance: as the content of a guideline is not static but may change over time, the representation must be able to store didactic and maintenance information such as author names, versioning information, purposes and detailed explanations.

The guideline model should be supported by a formal language (vocabulary, syntax and semantics) which specifies the actual guidelines in terms of the above-mentioned model constructs. Usually, such a language consists of two parts: a control-flow language and an expression language. The control-flow language usually specifies the guideline structure (flow) in terms of constructs of the model (e.g., the various tasks of the above-mentioned TNM), whereas the expression language usually describes decision criteria (e.g., 'is the patient older than 65 years'). This formal language (including both parts) must be interpretable by automatic parsers. Preferably, each approach should include a guideline execution engine, which incorporates such a parser that is able to provide decision support based on the encoded guidelines.

This chapter will compare and discuss a number of approaches in terms of these characteristics. The approaches discussed in this chapter are selected on the basis of information from OpenClinical [2] and the knowledge of the authors on existing approaches. Inclusion of an approach was based on the following criteria. First of all, as this chapter aims at identifying and discussing CIG approaches in terms of different characteristics (e.g., model, language), we selected approaches that differ as much as possible with respect to these characteristics. Furthermore, we used criteria such as lifetime of the approach and number of publications and whether the approach is considered generally as a 'key' approach. The final inclusion of an approach as a relevant subject was based on a subjective decision. Therefore, although we recognize that a number of other important approaches exist nowadays [2] such as PRODIGY, GUIDE, Gaston, GLARE, SAGE, HELEN, DeGel and SEBASTIAN, we have limited the number of refereed approaches (also to constrain the chapter's size) to the following five: The Arden Syntax [11], GLIF [12], PRO*forma*, [13], Asbru [14] and EON [15].

The remaining part of this chapter describes each of the five approaches, after which all approaches are compared in terms of the above-mentioned characteristics. The chapter finishes with a general discussion on guideline approaches, their strong points and their implications for future research.

1. The Arden Syntax

1.1. Introduction

Named after the Arden Homestead Conference Centre, where the initial meeting was held, the first version of the Arden Syntax was developed in 1989 [11] as a response to the inability to share medical knowledge among different institutions. The Arden Syntax 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) [16]. 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 [17]. 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 [18], 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.

1.2. Model

1.2.1. Medical Logic Modules—In the Arden Syntax, guidelines are modelled as (a collection of) Medical Logic Modules (MLMs). Each MLM represents a single decision and contains slots that are grouped into three categories: *Maintenance, Library* and *Knowledge*. The *Maintenance* and *Library* categories describe the MLM's pragmatics (e.g., title, version, explanation and keywords) and the *Knowledge* category describes the logic of an MLM. Figure 2 shows an example of (part of) an 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 an MLM in more detail.

1.2.2. Maintenance and Library Slots—As MLMs are to be shared among various institutions, the *Maintenance* and *Library* categories contain necessary documentation for each MLM. The *Maintenance* slots include the MLM's (file)name, author, version, institution, specialist, date of last modification and validation status (e.g., '*testing*', '*research*', '*production*' or '*expired*').

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).

1.2.3. 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*.

The *data* slot is used to obtain the values of concepts that are mentioned in the MLM from local clinical information systems such as Electronic Patient Record (EPR) systems. 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 figure 2) 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 [19].

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 figure 2 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).

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, list operators, temporal operators and aggregation operators. 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.

Once the logical expression evaluates to '*true*', the *action* slot is executed, performing whatever actions are specified in this slot. 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 figure 2 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).

The TNM of a single MLM always consists of three steps: the *evoke* slot determines whether the *logic* slot should be executed, which on its turn determines whether the *action* slot should be carried out. Although it is possible for an MLM to invoke other MLMs by means of the '*call*' statement in the *action* slot, the approach does not support a formal TNM to steer these invocations [20].

1.3. Language

The Arden syntax TNM is formally defined in Backus-Naur Form (BNF). MLMs are text-based (each MLM is encoded as an ASCII file) and always have the format, shown in figure 2. The expression language encodes criteria (in the *logic* slot) as textual production rules.

The approach does not contain a standard execution engine that is able to interpret and execute guidelines. However, a number of implementations for executing MLMs have been developed, including the use of pseudocode [21], C++ [22] and MUMPS [23]. As the Arden Syntax leaves the implementation of patient data modelling entirely up to the local institutions, there are no standard mapping facilities to obtain values of required patient data during guideline execution.

2. The GuideLine Interchange Format (GLIF)

2.1. Introduction

The GuideLine Interchange Format (GLIF) was developed to model guidelines in terms of a flowchart that consists of structured scheduling steps, representing clinical actions and decisions. Figure 3 shows an example of a GLIF guideline (aimed at the treatment of chronic cough), visualised through the Protégé knowledge modelling tool [24].

GLIF was developed by the Intermed Collaboratory [25] including researchers at Columbia University, Harvard University and Stanford University and was first published in 1998 [12]. The intended purpose of GLIF is to facilitate sharing of guidelines between various institutions by modelling 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. The current version of GLIF is GLIF3 [26], which is discussed in the remaining part of this section.

A variety of guidelines [8, 27, 28] have been specified using GLIF to evaluate the various aspects of the approach.

2.2. Model

2.2.1. Multi-level Approach—In order for guidelines to be 1) readable by humans, 2) interpretable by computers and 3) adaptable by different (local) institutions, GLIF defines 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, the guideline content is formally defined and various verification checks of the guidelines are carried out (this level is described in more detail in section 2.2.2). 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 (see also section 2.2.3).

2.2.2. The Computable Level—In contrast to the conceptual level, where guidelines are visualised merely as graphical flowcharts, the computable level allows for a formal specification, using the GLIF TNM. This model is object-oriented and consists of a number of classes that describe typical guideline tasks (e.g., decisions and actions).

The *Guideline* class represents a (sub)guideline. Each guideline is modelled as an instance of this class. The *Guideline* class 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). In addition, it also contains a reference to a collection of steps that are linked together in a directed graph (flowchart). Similar to a guideline, steps are also represented by classes, and each step in a guideline is an instance of such a class. GLIF defines five classes that represent the following steps: *Decision* steps, *Patient state* steps, *Branch* steps, *Synchronization* steps and *Action* steps.

Decision steps model decision points in a guideline and direct flow control from one guideline step to various alternatives. There are two types of *Decision* steps: *Case* steps and *Choice* steps. A *Case* step is a *Decision* step that contains a number of logical expressions and thus is used to model deterministic decisions. Based on the outcome, the guideline flow is directed to the various alternatives. In contrast, *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.

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* step 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.

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 certain branch 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.

Action steps model actions that have to (or should) be performed. Three types of actions 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 an MLM-macro, which can be used to define an MLM. Internally, the macro consists of two steps: a *Decision* step and an *Action* step.

2.2.3. The Implementable Level—Similar to the Arden Syntax, decision criteria and action specifications in GLIF contain references to actual patient data (e.g., the age of a patient) and medical 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, this information is stored in the implementable level. This level contains the information to integrate developed guidelines with institution-specific medical knowledge sources and information systems such as EPRs. GLIF aims at defining the structure of patient data elements and medical concepts in this level in accordance with standard data models and medical terminologies such as HL7's Reference Information Model (RIM) [29] or the Unified Medical Language System (UMLS) [30]. The implementable layer is currently being further developed [31].

2.3. Language

In GLIF, the guideline TNM itself (e.g., all classes, attributes and relations) is described by means of Unified Modeling Language (UML) class diagrams. The control-flow language that describes the actual guidelines (in terms of class instances) is the Resource Description Format (RDF) language.

Decision criteria are specified through a formal expression language, referred to as the Guideline Expression Language (GEL) [32], which is a superset of the Arden Syntax. In addition, the object-oriented expression language GELLO has been developed to specify decision criteria in GLIF [33]. 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 GELLO standard has recently been accepted as a standard expression language by HL7 and ANSI.

A guideline execution engine named GLEE (GuideLine Execution Engine) has been developed which is able to execute GLIF-encoded guidelines and can be integrated into the clinical information system of a local institution [34].

A separate approach is the development of a generic guideline execution engine, named the Guideline Execution by Semantic Decomposition of Representation (GESDOR) [35], which is able to execute various control-flow and expression languages, GLIF being one of them. The GLIF model and language are still being further developed.

3. PRO*forma*

3.1. Introduction

PRO*forma* is a CIG approach supported by acquisition and execution tools with the goal of supporting guideline dissemination in the form of decision support systems that assist patient care through active decision support and workflow management [13]. PRO*forma* was initially developed at the Cancer Research UK Advanced Computation Laboratory. The name PRO*forma* is a concatenation of the terms proxy ('authorised to act for another') and formalize ('give definite form to').

Similar to GLIF, PRO*forma* also represents guidelines as a directed graph in which the nodes are instances of a fixed set of classes. Figure 4 shows an example of a guideline in terms of instances of these classes, visualised through the Arezzo Composer, part of the Arezzo suite, developed by Infermed Ltd. [36].

Besides the commercially available Arezzo suite, a second suite has been developed to acquire and implement PRO*forma* guidelines, called the Tallis suite [37].

A large amount of CIG and CIG-based decision support systems have been developed in various areas by means of the Arezzo and Tallis suites [38].

3.2. Model

The PRO*forma* TNM is called the PRO*forma* task ontology. Each guideline in PRO*forma* is modelled as a plan that consists of a sequence of tasks. The PRO*forma* task ontology defines four classes, each with their own attributes: *Plans*, *Decisions*, *Actions* and *Enquiries*. These four tasks are derived from the generic *Keystone* task, which 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 and conditions.

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 *Keystone* task, the plan task contains additional attributes that store the plan's task network, scheduling and temporal constraints and abort or termination conditions.

The plan's network is stored as a set of task instances (similar to the *Guideline* class in GLIF). For example, a guideline that consists of four task instances (e.g., '*history*', '*diagnosis*', '*therapy*', '*follow-up*') is modelled through a *Plan* instance that 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 ten weeks*'). By using these two types of constraints, tasks in a plan are modelled differently than traditional flowcharts that order guideline elements usually only through scheduling constraints.

Another way of directing guideline flow in PRO*forma* is through abort or termination conditions. Each PRO*forma* 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.

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 [39] and support (+) or oppose (−) candidates, establishing a preference order among the candidates.

An *Action* is a task that a PRO*forma* 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 PRO*forma* 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 PRO*forma*, actions are always atomic and are not decomposable.

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.

3.3. Language

Guidelines in PRO*forma* are stored (as instances of the PRO*forma* task ontology) using a language, derived from the so-called Red Representation Language (R^2L), a time-oriented control-flow language [40]. In PRO*forma*, a guideline is a declarative specification of tasks and their (inter)relationships organised in a hierarchy of plans and their components. This language also contains a formal expression language to express goals (e.g., '*achieve(normal_respiration)*'), conditions (e.g., '*peak_flow < 30*', '*risk_level = severe*') and argument schemes ('*diagnosis = oesophagitis and liver_disease = absent then cimetidine*: +').

Before execution, guidelines 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 execution module.

Execution of PRO*forma* guidelines is supported by means of two execution engines, which are part of the earlier mentioned Arezzo and Tallis suites. In addition, the earlier-mentioned generic execution engine GESDOR [35] is also able to execute PRO*forma* guidelines.

4. Asbru

4.1. Introduction

Asbru is a CIG approach, developed at Stanford University, the Vienna University of Technology and the Ben-Gurion University, which focuses on the application and critiquing of time-oriented clinical guidelines [14]. This approach aims at representing 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. An example of an Asbru guideline can be shown in figure 4 of Chapter 8.

4.2. Model

The Asbru TNM represents guidelines as skeletal plans. Similar to the notion of plans in PRO*forma*, a plan is a collection of other (sub)plans and/or actions. The Asbru TNM consists of a number of elements, of which the *Plan* element is the most important one. Besides administrative attributes (e.g., the plan's title), each plan contains the following attributes (referred to as knowledge roles in Asbru), which describe each plan's functionality: *preferences*, *intentions*, *conditions*, *effects* and *plan body*.

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*').

Intentions are used to model the aims of the plan, independent of the plan body. Intentions can for example aid in the selection of the most appropriate plan by for example defining which patient state(s) must hold during of after a plan's execution (e.g., the patient's blood pressure must never exceed 140/90) or which actions should take place (e.g., maintain monitoring of blood glucose once a day). Intentions are modelled as temporal patterns.

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 defines 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.

Besides the earlier-mentioned *intentions* and *conditions*, *effects* can also be used to select the most appropriate plan by describing the expected behavior of the plan's execution. For example, a treatment plan might decrease the blood-glucose level, which classifies this plan as not the most appropriate for a certain class of patients. Effects may include probabilities that specify the probability of the effect's occurrence.

The *plan body* is a set of subplans or actions (which are plans that do not contain any subplans anymore) that have to be performed whenever the plan is considered appropriate (based on the plan's preconditions, intentions or effects). The order in which the subplans are executed is determined by the *Plan*'s *type* and *subtype* attributes. Asbru defines four types for synchronising subplans: '*sequentially*', '*parallel*', '*any-order*', and '*unordered*', which are described by means of the subplans' *type* attribute. Furthermore, cyclical plans can be defined. As mentioned earlier, plans that have been started can be suspended, aborted or completed (based on the plan's conditions).

4.3. Language

The Asbru TNM itself is defined as a Document Type Definition (DTD), which defines the structure of the various elements of the TNM. The control-flow language and the expression language are formally defined by means of XML, based on the earlier-mentioned DTD [41].

An important aspect of the expression language is the concept of time annotations, which are used in specifying complex temporal patterns, and specify the temporal constraints within which an action must take place, or a condition must be fulfilled in order to trigger. The Asbru expression language also supports the concept of temporal abstractions such as 'has the patient suffered from a second episode of anaemia of at least moderate severity'.

Beside the above-mentioned XML language, other languages, related to Asbru such as MHB (Many-Headed Bridge) are being developed with the goal of bridging the gap between informal (e.g., textual) and formal guidelines [42].

Various execution engines have been developed that are able to execute (a subset of) Asbru guidelines such as the Asbru interpreter [43], developed as part of the Protocure-II project [44] and the Asbru Execution Engine [45]. Another execution engine that has been developed to execute (simplified) Asbru guidelines is the Spock system [46], which is part of the DeGel framework [47].

5. EON

5.1. Introduction

EON, developed at Stanford University, is a CIG approach that aims at developing decision support systems that reason about guideline-directed care [15]. The EON approach consists of several components that facilitate the acquisition and execution of clinical guidelines.

Similar to GLIF, the EON TNM, called Dharma [48], is object-oriented and consists of classes that describe guideline tasks as a sequence of structured temporal steps. The Dharma model is non-monolithic, meaning that it can be extended with additional classes that capture new guideline behaviour. Besides the Dharma guideline model, the EON architecture also contains a number of run-time components, used to construct execution-time systems. An example of a guideline in EON, visualised (similar to GLIF guidelines) through the Protégé knowledge modelling tool [24] can be seen in figure 1 of chapter 8.

The EON project has been discontinued since 2002, but was succeeded by the SAGE project, which ran from 2002-2006 [49]. One of the CIG systems that were developed using EON is the ATHENA system, which addresses the treatment of hypertension, and is still in clinical use today [50].

5.2. Model

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

The EON approach aims at modelling multi-encounter patient management (e.g., chronic disease management), in which each guideline represents a certain state of the patient and consists of a number decisions and actions that are applicable to that patient state and may lead to changes in patient states over time (and may also trigger other guidelines as a result). In order to define guidelines according to this conceptual model, the TNM consists of a number of standard classes and attributes which form the so-called core guideline ontology. The four most important classes are *Scenarios*, *Decisions*, *Actions* and *Activities*.

A *Scenario* is a (partial) characterisation 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 –which were firstly introduced as part of the PRODIGY guideline approach [51]– 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 and 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.

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 alternatives (similar to the concept of schemas in PRO*forma*).

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 modelled 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 synchronisation constructs in order to execute parallel tasks.

Every class in the Dharma ontology can be associated with a goal. The notion of goals is comparable with the notion 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 three weeks*’).

5.2.2. The Patient Data and Medical-specialty Model—The patient data model defines classes and attributes in order to represent patient data. It defines characteristics regarding demographic and clinical conditions of specific patients. It does not aim at modelling 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.

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 organised classes, relations and attributes. The medical-specialty model represents different sorts of domain-specific information.

5.3. Language

The Dharma TNM as well as the control-flow language is described by means of the internal frame-based Resource Description Format (RDF) of Protégé. Although the focus of the EON approach is not on defining a formal control-flow language, it does particularly address the subject of defining criteria in formal expression languages. EON defines three different expression languages.

First, common but relatively simple criteria can be expressed as Boolean criteria in terms of a set of object templates such as ‘*diabetes mellitus is present and the most recent serum creatinine is less than normal*’.

According to the researchers of the EON approach, such a criterion language is not expressive enough to capture more complex criteria such as ‘*is an authorised 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.

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 two weeks after the initiation of lisinopril*’. These are written as temporal queries, which during guideline execution are translated to database queries. To specify temporal aspects, EON has adopted a subset of the Asbru temporal expression language to represent temporal information.

To facilitate the development of guideline execution engines, EON defines an extensive execution architecture that contains a guideline execution engine plus components for interfacing to third-party information systems [52].

6. Discussion

6.1. Comparison

6.1.1. Overview—Each approach focuses on different aspects of guideline modelling and representation, which have their implications regarding the representation and modelling of guidelines as shown in the previous sections. This section will address the strong points of each approach, after which a number of (minimal) requirements are formulated that were distilled from these points that can be used in the process of developing new approaches or improving existing ones.

6.1.2. Model—All approaches described in this chapter use a TNM that models guidelines in terms of a sequence of class instances (e.g., flowchart), with the exception of the Arden Syntax that models guidelines as (a collection of) independent modular rules. As a result, the Arden Syntax is most suitable for representing simple guidelines such as alerts in reminder systems, but less suitable for complex multistep guidelines.

Although the terminology may differ, all approaches support a basic set of ‘core’ guideline tasks, such as decisions, actions and entry criteria. Decisions for example are represented by means of *logic* slots in the Arden Syntax, *Decision* steps in GLIF, *Decision* tasks in PRO*forma, conditions* in Asbru, and *Decisions* in EON. Similarly, actions are represented by means of *action* slots in the Arden Syntax, *Action* steps in GLIF, *Action* and *Enquiry* tasks in PRO*forma, actions* (atomic *Plans*) in Asbru, and *Actions* in EON. Entry criteria are represented by *evoke* slots in the Arden Syntax, *Patient state* steps in GLIF, *preferences, intentions* and *effects* in Asbru, *triggers* and *conditions* in PRO*forma* [53] and *Scenarios* in EON.

The TNMs of all approaches define a fixed set of guideline tasks, with the exception of EON that is extensible.

All approaches described in this paper except for the Arden Syntax provide explicit support for controlled 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* task 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 supports a form of nesting by calling other rules in the *Action* slot, there is no general control flow that controls these invocations. All approaches support the concept of referenced subguidelines. 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).

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 intention model is the most sophisticated. GLIF defines different layers of abstraction, which allows guideline authors to view only the general control flow (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.

Besides the knowledge that defines the guideline control flow (in terms of for example, rules, steps, plans), 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 an MLM. As a result, an MLM does not 'know' for example that amoxicillin is an antibiotic. Also PRO*forma* and Asbru contain no explicit support for modelling domain-specific knowledge or for using standard terminology systems. GLIF addresses this problem by modelling domain-specific knowledge through the implementable level and EON takes a similar approach by defining, the Patient Data and the Medical-Specialty models.

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.

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).

6.1.3. Language—All approaches define a control-flow language that describes the guideline control flow (in terms of TNM constructs). All approaches except EON have also defined the TNM in a formal way using BNF, XML or UML. For each approach, the control-flow language supports all constructs of the corresponding TNM. The Arden Syntax describes guidelines using formatted text, GLIF and EON using RDF, Asbru using XML and PRO*forma* using R^2L. PRO*forma* is the only approach, which makes a distinction between a declarative language (R^2L), used during the guideline definition phase and a procedural language (L_{R2L}) that is processed during the guideline execution phase. In order to facilitate this translation, 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 conditions). Similar to PRO*forma*, Asbru also uses the concept of guideline execution states.

All approaches also define formal expression languages that describe decisions and entry criteria. An important aspect of these languages is the issue of temporal reasoning. 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. EON expresses criteria using a description that is very similar to that of GLIF, with the main exception that GLIF describes expressions in GEL/GELLO while EON describes expressions by means of the three different criterion languages. 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. As many guidelines (especially treatment guidelines) are rather deterministic by nature, the issue of representing temporal aspects seems to have higher priority that the issue of representation of uncertainty (although this might be less true for diagnostic guidelines).

All approaches have developed execution engines 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 task). The Arden syntax, PRO*forma* and EON have published results on the development and implementation of actual decision support systems in daily care. PRO*forma* is the only approach described here that has developed a commercialised version. PRO*forma*, EON and Asbru execution engines are able to communicate with clinical information systems and users through standard Application Programming Interfaces (APIs) or communication protocols (e.g., web services).

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.

6.2. Requirements

The descriptions and comparisons in the previous sections show that each approach has a number of strong points. This section formulates requirements that were distilled from these points that can be used in the process of developing new approaches or improving existing ones.

6.2.1. Model—A guideline TNM must contain a set of generic guideline tasks that is able to represent all facets of simple as well as complex diagnostic and treatment guidelines. This set must be understandable on a functional level by guideline authors and on an executable level by computerised decision support systems.

A guideline TNM must support at least the two necessary basic tasks: actions and decisions. In order to be able to specify guideline-oriented actions (e.g., ‘*prescribe new medication*’ or ‘*diagnose patient with hypertension*’) as well as programming-oriented actions (e.g., ‘*get all drugs from an EPR*’ or ‘*give message to user*’) a guideline TNM must 1) provide a very expressive and rich model that enables the specification of all above-mentioned actions in a limited set of tasks or 2) provide the ability to derive new (sub)tasks from the existing ones that define new functionality.

Other important tasks in a guideline TNM are tasks that influence guideline flow such as entry/exit points (e.g., *Patient state* steps) and repetition/loops (e.g., *Synchronization* steps or the Asbru *Plan type*).

The guideline TNM must be able to represent various kinds of guidelines, that may differ considerably in complexity in a consistent manner such as relatively simple guidelines that model independent modular rules (e.g., MLMs in the Arden Syntax or MLM-macros in GLIF), but also complex guidelines such as clinical trials or treatment plans. In order to represent these various types of guidelines in a consistent manner, the approach must be able to represent guidelines on multiple levels of abstraction such as nesting, task or guideline decomposition (e.g., subguidelines or subplans in GLIF, EON, PRO*forma* and Asbru), and specifying the guideline’s intention or goal (e.g., Asbru’s intentions).

CIGs that are used for active decision support must be integrated with existing clinical information systems such as EPR systems. Concepts that are used in a guideline such as patient demographics, results of laboratory tests, indications and drugs must be explicitly defined so that they can be mapped to entries in a clinical information system. To facilitate the (re)use of a guideline among different institutions and systems, the reasoning knowledge (e.g., the used methods or tasks) must be separated from domain-specific knowledge (e.g., used drugs or laboratory tests). Also, the representation should support the use of standard data models and medical terminologies such as HL7, UMLS and SNOMED (e.g., the multi-level approaches in GLIF and EON).

Furthermore, in order to further facilitate the sharing of guideline-based decision support systems and to increase the acceptance of (national) guidelines in local institutions, actions that are programming-related must be separated from actions that are not. In this manner, institution-specific actions (e.g., sending an email to a physician vs. showing a message on a screen) are defined separate from the knowledge that describes the guideline itself. For example, guidelines may contain an additional ‘layer’ that describes such actions, independent of the guideline process. This is supported by GLIF and EON as it is possible to describe multiple kinds of tasks for each action such as decision support-related or programming-related tasks.

A guideline representation must be able to hold didactic- and maintenance-related information such as author names, versions, (literature) references, sources and referees. Especially versioning-related information is very important, as guidelines are usually dynamic (the contents may change rapidly over time) and national guidelines may be adapted to local institutions.

6.2.2. Language—A CIG approach should define formal control-flow languages as well as expression languages that are able to capture all the requirements mentioned above, in an unambiguous way. On the one hand, these languages must be abstract enough so that it is interpretable by guideline acquisition/visualisation tools (e.g., Protégé, Arezzo Composer) and guideline authors who do not have a logical or modelling background are able to define the process (e.g., flow), decision criteria and actions in a guideline (a more detailed description on various guideline acquisition/visualisation tools can be found elsewhere [3] and in Chapter 8: Visualization Methods to Support Guideline-Based Care Management). On the other hand, the languages must be interpretable (and preferable also verifiable) by engines that are able to execute guidelines. Such an execution engine must be able to interface with various clinical information systems in a consistent manner, for example by mapping concepts from the guideline to corresponding items in a clinical information system (e.g., the concept *Drug* in a guideline must be mapped to a drug table of an information system’s database). Also, actions that a guideline performs must be configurable as they may differ in various local situations (e.g., send an e-mail in a certain situation in contrast to issuing an on-screen alert in another one). This implies a component-based approach in which each component performs a specific task such as reasoning or interfacing. The encoded format as well as the guideline execution engine must meet execution-time requirements such as compactness and execution speed.

Temporal logic is a very important issue in guideline modelling. Guidelines usually refer to complex temporal constructs to describe for example drug prescription schemes. Therefore, a guideline representation model must contain an expressive means of modelling temporal expressions (e.g., Asbru’s temporal logic). The truth-value of a decision can not always be evaluated as ‘*true*’ or ‘*false*’, for example in the case of missing data (e.g., the patient’s medical history is not known). Guideline models must be able to handle such situations (e.g., using the relatively simple three-valued logic in GLIF or the more complex R2L language in PRO*forma*).

7. Research Agenda

In the last decade, most of the attention on computer-based guideline development has been focused on the areas of guideline representation models and underlying languages. From this research, various approaches and models arose, each with its own focus points and related strengths and weaknesses. However, based on the comparison in this chapter as well as looking at other studies [3, 4, 5-8], the conclusion can be drawn that the minimal necessary components for guideline representation have been identified. The next step in this process will be to develop a standard guideline representation model using these components (e.g., under the HL7 auspices [54]), keeping in mind the main conclusion that was drawn by Peleg et al.: '*…that because of the different goals of various research groups, a consensus model will be acceptable to the research groups only if it concurrently allows them to continue their investigations of unique features*' [8].

Also, the real benefit lays in structuring and guiding the whole guideline development process: in order to successively computer-based guideline systems that will be used in daily practice, various aspects such as representation, acquisition, verification and execution must be taken into account (a nice example of such an approach is the DeGel project from the Ben-Gurion university [47]). This is not a trivial task. Comparing the various approaches shows that design specifications made in the area of guideline representation have implications in the area of guideline execution (e.g., the 'fuller' the language, the less executable it will be).

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 (e.g., the GESDOR approach). Various solutions may be developed that address these issues such as the development of versioning methods that enable synchronisation between national and local guidelines and the development of standard interfaces to different external information systems. Recently, a number of articles have been published that compare CIG approaches with traditional workflow languages [6]. We expect that these comparisons will contribute significantly to the process of defining requirements and standards for CIG approaches (see also Chapter 3: From guidelines to careflows: modelling and supporting complex clinical processes).

In order to create an approach that is successful, it is important that future research will take into account that 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, formalisation, acquisition and execution.

References

[1]. James BC. Making it easy to do it right. N Engl J Med. 2001; 345(13):991–3. [PubMed: 11575294]

[2]. OpenClinical. Accessed March 11, 2008Available at: http://www.openclinical.org.

[3]. De Clercq PA, Blom JA, Korsten HH, Hasman A. Approaches for creating computer-interpretable guidelines that facilitate decision support. Artif Intell Med. May; 2004 31(1):1–27. [PubMed: 15182844]

[4]. Peleg, M. Guideline and Workflow Models. In: Greenes, Robert A., editor. Medical Decision-Making: Computational Approaches to Achieving Healthcare Quality and Safety. Elsevier/ Academic Press; 2006.

[5]. Tu S, Musen M. Representation Formalisms and Computational Methods for Modeling Guideline-Based Patient Care. First European Workshop on Computer-based Support for Clinical Guidelines and Protocols. 2000:125–42.

[6]. Mulyar N, van der Aalst WM, Peleg M. A pattern-based analysis of clinical computer-interpretable guideline modeling languages. J Am Med Inform Assoc. Nov-Dec;2007 14(6):781–7. [PubMed: 17712087]

[7]. 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 Inform. Dec 18; 2002 68(1-3): 59–70. [PubMed: 12467791]

[8]. 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. Jan-Feb;2003 10(1):52–68. [PubMed: 12509357]

[9]. De Clercq PA, Hasman A. Experiences with the development, implementation and evaluation of automated decision support systems. Medinfo. 2004; 11(Pt 2):1033–7.

[10]. Elkin PL, Peleg M, Lacson R, Bernstam E, Tu S, Boxwala A, Greenes R, Shortliffe EH. Toward Standardization of Electronic Guideline Representation. MD Computing. 2000; 17(6):39–44. [PubMed: 11189760]

[11]. 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. IEEE Computer Society Press; New York: 1989. p. 116-21.

[12]. 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. [PubMed: 9670133]

[13]. Fox J, Johns N, Rahmanzadeh A. Disseminating medical knowledge: the PRO*forma* approach. Artif Intell Med. 1998; 14:157–81. [PubMed: 9779888]

[14]. 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. [PubMed: 9779882]

[15]. Musen MA, Tu SW, Das A, Shahar Y. EON: A Component-Based Approach to Automation of Protocol-Directed Therapy. JAMIA. 1996; 3:367–88. [PubMed: 8930854]

[16]. Hripcsak G, Ludemann P, Pryor TA, Wigertz OB, Clayton PD. Rationale for the Arden Syntax. Comput Biomed Res. 1994; 27(4):291–324. [PubMed: 7956129]

[17]. 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. Jun; 2001 34(3): 170–81. [PubMed: 11723699]

[18]. Clinical Decision Support & Arden Syntax Technical Committee of HL7. inventor Arden Syntax for Medical Logic Systems. version 2.0. USA: Jul 7. 1999 Draft revision

[19]. Pryor TA, Hripcsak G. Sharing MLMs: an experiment between Columbia-Presbyterian and LDS Hospital. Proc Annu Symp Comput Appl Med Care. 1993:399–403. [PubMed: 8130503]

[20]. 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. Proc Annu Symp Comput Appl Med Care. 1995:238–42. [PubMed: 8563276]

[21]. Hripcsak G, Cimino JJ, Johnson SB, Clayton PD. The Columbia-Presbyterian Medical Center decision-support system as a model for implementing the Arden Syntax. Proc Annu Symp Comput Appl Med Care. 1991:248–52. [PubMed: 1807598]

[22]. 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. [PubMed: 7705067]

[23]. McCauley B, Young I, Clark I, Peters M. Incorporation of the Arden Syntax within the reimplementation of a closed-loop decision support system. Comput Biomed Res. Dec; 1996 29(6):507–18. [PubMed: 9012572]

[24]. Noy NF, Crubezy M, Fergerson RW, Knublauch H, Tu SW, Vendetti J, Musen MA. Protégé-2000: an open-source ontology-development and knowledge-acquisition environment. AMIA Annu Symp Proc. 2003:953. [PubMed: 14728458]

[25]. Peleg M, Boxwala AA, Tu S, Zeng Q, Ogunyemi O, Wang D, Patel VL, Greenes RA, Shortliffe EH. The InterMed approach to sharable computer-interpretable guidelines: a review. J Am Med Inform Assoc. Jan-Feb;2004 11(1):1–10. [PubMed: 14527977]

[26]. Boxwala AA, Peleg M, Tu S, Ogunyemi O, Zeng QT, Wang D, Patel VL, Greenes RA, Shortliffe EH. GLIF3: a representation format for sharable computer-interpretable clinical practice guidelines. J Biomed Inform. Jun; 2004 37(3):147–61. [PubMed: 15196480]

[27]. Choi J, Currie LM, Wang D, Bakken S. Encoding a clinical practice guideline using guideline interchange format: A case study of a depression screening and management guideline. Int J Med Inform. Oct; 2007 76(Suppl 2):S302–7. [PubMed: 17600762]

[28]. Wang D. Translating Arden MlMs into GLIF guidelines--a case study of hyperkalemia patient screening. Stud Health Technol Inform. 2004; 101:177–81. [PubMed: 15537224]

[29]. Schadow G, Russler DC, Mead CN, McDonald CJ. Integrating Medical Information and Knowledge in the HL7 RIM. Proc AMIA Symp. 2000:764–8. [PubMed: 11079987]

[30]. Lindberg C. The Unified Medical Language System (UMLS) of the National Library of Medicine. J Am Med Rec Assoc. 1990; 61(5):40–2. [PubMed: 10104531]

[31]. Peleg M, Keren S, Denekamp Y. Mapping computerized clinical guidelines to electronic medical records: Knowledge-data ontological mapper (KDOM). J Biomed Inform. Feb; 2008 41(1):180–201. [PubMed: 17574928]

[32]. 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. Proc AMIA Symp. 2001:523–7. [PubMed: 11825243]

[33]. Sordo M, Ogunyemi O, Boxwala AA, Greenes RA. GELLO: An Object-oriented Query and Expression Language for Clinical Decision Support. Proc AMIA Symp. 2003:1012. [PubMed: 14728515]

[34]. Wang D, Shortliffe EH. Design and implementation of the GLIF3 guideline execution engine. J Biomed Inform. Oct; 2004 37(5):305–18. [PubMed: 15488745]

[35]. 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. Proc AMIA Annual Symp. 2003:694–698.

[36]. InferMed Ltd. Accessed March 11, 2008Available at http://www.infermed.com

[37]. Accessed March 11, 2008The Tallis Toolset. Available at http://www.acl.icnet.uk/lab/tallis/index.html

[38]. Fox J, Patkar V, Thomson R. Decision support for health care: the PROforma evidence base. Inform Prim Care. 2006; 14(1):49–54. [PubMed: 16848966]

[39]. Fox, J.; Das, S. Arguments about beliefs and Actions: Decision making in the Real World. In: Fox, J.; Das, S., editors. Safe and Sound: Artificial Intelligence in Hazardous Applications. 2000. p. 55-76.

[40]. 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.

[41]. Accessed March 11, 2008Asbru syntax. Available at http://www.asgaard.tuwien.ac.at/plan_representation/asbrusyntax.html

[42]. Seyfang A, Miksch S, Marcos M, Wittenberg J, Polo-Conde C, Rosenbrand K. Bridging the Gap between Informal and Formal Guideline Representations. Proc. 17th European Conference on Artificial Intelligence. 2006:447–51.

[43]. Votruba, P.; Seyfang, A.; Paesold, M.; Miksch, A. Improving the Execution of Clinical Guidelines and Temporal Data Abstraction in High-Frequency Domains, AI Techniques in Healthcare: Evidence-based Guidelines and Protocols. European Coordinating Committee for Artificial Intelligence; 2006. p. 112-6.

[44]. Accessed March 11, 2008Protocure II. Available at http://www.protocure.org

[45]. Miksch S, Fuchsberger C. Asbru's Execution Engine: Utilizing Guidelines for Artificial Ventilation of Newborn Infants. Proc. IDAMAP. 2003; S:119–25.

[46]. Young O, Shahar Y, Liel Y, Lunenfeld E, Bar G, Shalom E, Martins SB, Vaszar LT, Marom T, Goldstein MK. Runtime application of Hybrid-Asbru clinical guidelines. J Biomed Inform. Oct; 2007 40(5):507–26. [PubMed: 17276145]

[47]. Shahar Y, Young O, Shalom E, Galperin M, Mayaffit A, Moskovitch R, Hessing A. A framework for a distributed, hybrid, multiple-ontology clinical-guideline library, and automated guideline-support tools. J Biomed Inform. Oct; 2004 37(5):325–44. [PubMed: 15488747]

[48]. Tu SW, Musen MA. A flexible approach to guideline modeling. Proc AMIA Symp. 1999:420–4. [PubMed: 10566393]

[49]. Tu SW, Campbell JR, Glasgow J, Nyman MA, McClure R, McClay J, Parker C, Hrabak KM, Berg D, Weida T, Mansfield JG, Musen MA, Abarbanel RM. The SAGE Guideline Model: achievements and overview. J Am Med Informtgt Assoc. Sep-Oct;2007 14(5):589–98.

[50]. Goldstein MK, Coleman RW, Tu SW, Shankar RD, O'Connor MJ, Musen MA, Martins SB, Lavori PW, Shlipak MG, Oddone E, Advani AA, Gholami P, Hoffman BB. Translating research into practice: organizational issues in implementing automated decision support for hypertension in three medical centers. J Am Med Inform Assoc. Sep-Oct;2004 11(5):368–76. [PubMed: 15187064]

[51]. Johnson PD, Tu S, Booth N, Sugden B, Purves IN. Using scenarios in chronic disease management guidelines for primary care. Proc AMIA Symp. 2000:389–93. [PubMed: 11079911]

[52]. Tu SW, Musen MA. Modeling Data and Knowledge in the EON Guideline Architecture. Medinfo. 2001; 10:280–4.

[53]. Bury JP, Saha V, Fox J. Supporting 'scenarios' in the PROforma guideline modeling format. Proc AMIA Symp. 2001:870–4.

[54]. Jenders RA, Sailors RM. Convergence on a standard for representing clinical guidelines: work in health level seven. Medinfo. 2004; 11(Pt 1):130–4.

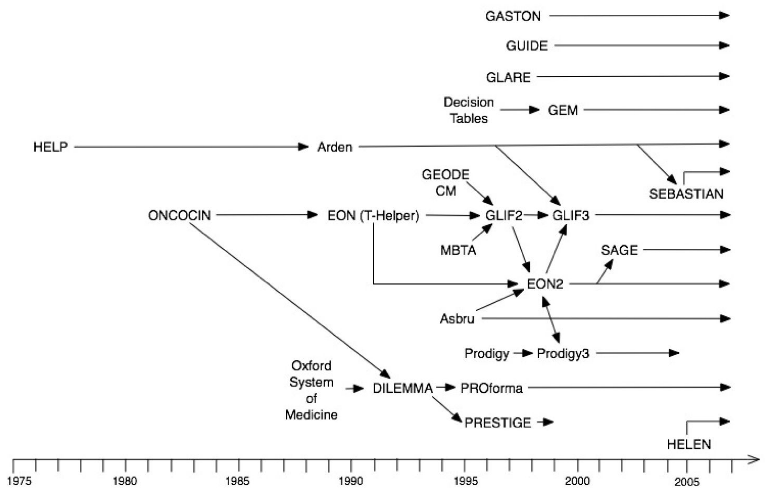

Figure 1.
History of CIG approaches, positioned on a time axis (adapted from Elkin et al [10])

```
maintenance:
  title: Alert on low hematocrit;;
library:
  purpose: Warn provider of new or worsening anemia.;;
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:
    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 rapidly.";;
end:
```

Figure 2.
An example of an MLM

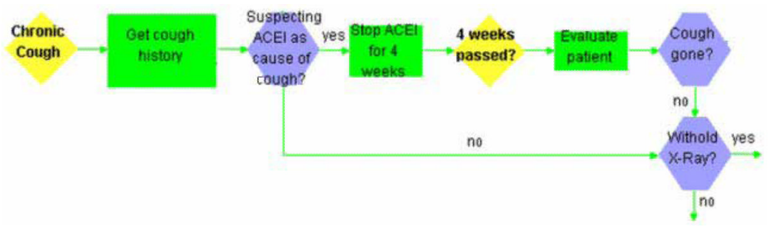

Figure 3.
Graphical representation of a treatment chronic cough treatment guideline in GLIF (adapted from Boxwala et al [26])

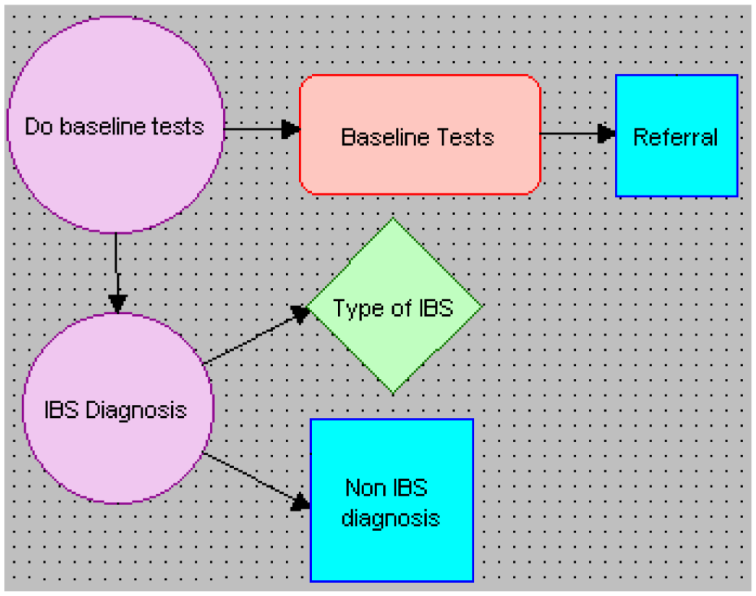

Figure 4.
Part of a PRO*forma* guideline