Using Features of Arden Syntax with Object-Oriented Medical Data Models for Guideline Modeling

Mor Peleg, Ph.D.[1], Omolola Ogunyemi, Ph.D.[2], Samson Tu, M.S. [1], Aziz A. Boxwala, M.B.B.S., Ph.D.[2], Qing Zeng, Ph.D.[2], Robert A. Greenes, M.D., Ph.D.[2], Edward H. Shortliffe, M.D., Ph.D.[3]

[1]Stanford Medical Informatics, Stanford University School of Medicine, Stanford, CA [2]Decision Systems Group, Harvard Medical School, Brigham & Women's Hospital, Boston, MA; [3] Department of Medical Informatics, Columbia University, New York, NY

Computer-interpretable guidelines (CIGs) can deliver patient-specific decision support at the point of care. CIGs base their recommendations on eligibility and decision criteria that relate medical concepts to patient data. CIG models use expression languages for specifying these criteria, and define models for medical data to which the expressions can refer. In developing version 3 of the GuideLine Interchange Format (GLIF3), we used existing standards as the medical data model and expression language. We investigated the object-oriented HL7 Reference Information Model (RIM) as a default data model. We developed an expression language, called GEL, based on Arden Syntax's logic grammar. Together with other GLIF constructs, GEL reconciles incompatibilities between the data models of Arden Syntax and the HL7 RIM. These incompatibilities include Arden's lack of support for complex data types and time intervals, and the mismatch between Arden's single primary time and multiple time attributes of the HL7 RIM.

1 Introduction

Computer-interpretable guidelines (CIGs) that are linked to electronic medical records (EMRs) are a means for providing patient-specific advice automatically at the point-of-care. There are several methodologies for encoding guidelines to make them computer-interpretable.[1-4] All of these methodologies have constructs for defining criteria that relate medical concepts to patient data. While each methodology has different constructs, they all use some sort of *expression language* for specifying logical decision and eligibility criteria, and a *data model* for medical concepts and patient data.

GLIF is a format for encoding and sharing computer-interpretable clinical guidelines developed by the InterMed Collaboratory, a joint project of medical informatics groups at Harvard, Stanford, and Columbia universities. GLIF3[4] is GLIF's third version. This paper discusses the approach that InterMed has used to design a data model and expression language for GLIF3.

GLIF3 is designed to support computer-based execution, with integration into clinical systems environments and access to data in the EMR. While the control-flow of the GLIF model was already formally expressed in GLIF2, decision and eligibility criteria, and patient data, were expressed in natural language that did not support execution.[4] Therefore, our goals for GLIF3 were to develop: (1) a formal expression language that allows computation on time-stamped data, and (2) a data model that specifies a hierarchy of classes, which represent medical concepts and data and defines their characterizing attributes, including timing aspects.

GLIF3 enables a guideline specification to be computable by formally specifying patient data items, action specifications, decision criteria, and eligibility criteria. For patient data items, there must be a definition of medical concepts to which the data refer. This definition is given by a *medical ontology*, which contains the clinical meaning of the concepts as well as their structure, or *data model.* As an example, a test result should have at least two attributes, one that represents the test value and one for the biologically-relevant time of the test. Defining medical concepts in relation to standard medical vocabularies allows the guideline encoding to contain concepts that are institution-independent. Mapping to institution-dependent EMR codes and procedures can therefore be specified in another level.

The GLIF3 guideline model uses the medical ontology in two ways. First, action specifications such as recommended test orders or notifications are defined in terms of formal medical concepts contained in the medical ontology. Second, a formal expression language uses data item names that are defined by the medical ontology. The expression language is used to express decision criteria, eligibility criteria, and patient states.

2 Background

In creating GLIF3, we have tried to build on standards so as to avoid duplicating the work of others.

Toward this end, we determined that working with the HL7 organization would be desirable. The Clinical Guidelines Special Interest Group in HL7 was formed this year, to move toward a standardized representation for computer-based guidelines. To further the harmonization with ongoing HL7 efforts, we developed an expression language derived from Arden Syntax's logic grammar. Arden Syntax is a standard of the American Society for Testing and Materials[6] and is being further developed under HL7. We used standard controlled medical vocabularies to define medical concepts. We investigated the HL7 RIM version 1.0 as a default medical data model, as GLIF3 allows a guideline encoder to use any object-oriented (OO) model for defining the structure of the concepts used by the guideline.

2.1 HL7's Reference Information Model

The clinical part of HL7's RIM version 1.0[5] can model medical knowledge and patient data uniformly. All clinical data are specified as *Act* objects, or *Acts*. Acts have attributes that provide information about clinical concepts they represent and their timing. Relationships are used to model the Act circumstances and allow grouping the Acts and reasoning about them. Acts are specialized into procedures performed on a patient, observations about the patient, medications given to the patient, and other kinds of services. Different sub-classes contain additional attributes that help characterize an Act. For example, Observation has a *value* attribute, whereas Medication has attributes about dosage and route of administration. Act objects have a *mood* that distinguishes the ways in which they can be conceived: as an event that occurred, a definition, intent, order, etc.[5]

2.2 Arden Syntax

Arden Syntax is used to compose Medical Logic Modules (MLMs), each of which represents a single medical decision. Arden Syntax assumes a simple data model. It supports the following simple types: null, Boolean, number, time (timestamp), duration, and string. In addition, it supports a list of simple types and a query result, which is a list whose values are associated with a primary time – the medically relevant time of occurrence. An element of a list or a query result can have only one data value; no complex data values are supported. For example, a drug's administration route and dose must be represented as different variables.

An MLM has logic and data slots that define its decision logic and the medical data items it uses. These slots use expressions whose syntax is defined by a grammar that we refer to as "Arden Syntax logic grammar". The data slot of an MLM in Arden Syntax specifies how data are retrieved from an EMR. However, only part of this specification is defined by the syntax. Mapping the MLM variables to the institutional EMR must be defined in a non-standard way within curly braces, a difficulty that is called the *curly braces problem* in the Arden community. This problem implies that sharing an MLM among different institutions requires adapting the MLMs to local systems and terminology standards.

3 GLIF3's expression language and its interaction with the medical data model

We used Arden Syntax's logic grammar as our basis for developing GLIF3's expression language, as Arden Syntax has been accepted as a standard for delivering alerts and reminders and is a mature language that has been implemented and tested by multiple academic groups and commercial vendors.

In developing GLIF3's domain ontology, we used an OO data model because of the advantages it provides. One advantage is that an OO data model encapsulates the attributes of a patient data item into a single complex data object. For example, an HL7 Medication object can specify the particular medication used, the dose, the route of administration, the time period for which it was prescribed, and the date of first prescription, among other things. Encapsulation improves data integrity and makes conceptualizing easier. Another advantage of an OO data model is that it enables a declarative way of specifying medical concepts and data items used by a guideline. This method could facilitate mapping concepts and data items to institutional EMRs, thus addressing Arden's curly braces problem, as suggested by Jenders et al [7].

HL7 RIM is suitable as a medical data model for GLIF3 because it is general enough to represent the data structures for a wide range of medical data and concepts in a uniform manner while using a small number of classes.[5] Moreover, HL7 is creating standards for messaging interfaces for EMRs based on the RIM.

Figure 1 shows an example of the definition of a GLIF3 data item representing an ACE Inhibitor treatment that uses the HL7 RIM Medication object.

3.1 Incompatibilities between Arden Syntax logic grammar and the HL7 RIM

Using Arden Syntax's logic grammar in the context of a complex OO data model is not straightforward, as there are several incompatibilities between the two models. As mentioned, the major incompatibility is that Arden Syntax has a simple data model, whereas the HL7 RIM model is object oriented. Thus, many

of the standard operators for Arden Syntax's logic grammar cannot be used compatibly with the HL7 RIM.

Another incompatibility is that Arden's data model does not contain intervals as a data type, whereas the data model in the HL7 RIM and in GLIF3 allows expression of temporal and other types of intervals. In addition, HL7 RIM data model classes can have many different times associated with them, while Arden Syntax is limited to a single timestamp. For example, in HL7 RIM, an Act data model class may be associated with an *activity time interval* that represents the time that the medical action took place, a *critical time interval* that represents the biologically relevant time period for the action, and a *recording (availability) timestamp* of the action. Arden Syntax defines many operators (e.g., *latest*) that assume that each data value can be associated with at most one time stamp. Clearly, this compatibility issue between the two models is problematic.

```
(Instance of Variable_Data_Item)
   {name: ACEI_Item
    concept:    {(instance of Concept)
                   concept_name: ACEI;
                   concept_id: C0003015;
                   concept_source: UMLS}
    data_model_class_id: Medication
    data_model_source_id: HL7-RIM
    data_value: {(instance of Medication)
                   service_cd: ACEI concept;
                   mood_cd: event;
                   critical_time: {low: null;
                                   high: null;}…} }
```

Figure 1. A variable data item that defines ACE Inhibitor treatment. Attribute names are on the left, followed by their values. Complex values are in curly braces. The ACEI data specify the appropriate UMLS code and HL7 RIM class (Medication). The figure shows two attributes of Medication. Other attributes include dosage_quantity, rate_quantity, and route_code.

3.2 The GLIF3 approach for using Arden Syntax's logic grammar with the HL7 RIM

To address the incompatibilities, we created an expression language that we call the *Guideline Expression Language* (GEL). GEL is based on Arden Syntax's logic grammar but supports (1) complex data structures; (2) timestamp, numeric, and duration intervals; and (3) function definitions. In addition, we created constructs in GLIF3 that extract patient data taken from the EMR as HL7 RIM objects and transform them into GEL-compatible data structures. This solution allows an Arden-derived expression language to be used with an object-oriented data model. Examples of this approach follow.

3.2.1 GEL

In addition to supporting Arden Syntax's basic data types, GEL supports complex data types, intervals, lists of these new types, and query results, the data values of which can be a complex type or an interval. A complex data type can have any number of attributes, each of which is a GEL data type.

GEL supports most of the Arden Syntax logic grammar operators[1], and it also provides a mechanism for users to define functions. This feature is important for facilitating GLIF3's support for user-defined data model classes and instances. A user may want to define a data model in which primary time has no special meaning, but in which other meaningful relationships might be semantically important. For example, inheritance relationships among concepts (e.g., drug hierarchies) might be a central part of the data model.

GEL supports complex data types, but their attributes are treated equally and do not convey any special semantic meaning. A way to restore such semantics is to define GEL functions. For example, a function can be defined that would deduce the children of a concept based on inheritance relationships. This approach would restore the semantics of a "parent" attribute of a concept class.

Adding several functions that can be used by GLIF3 when writing GEL expressions is useful. The most important of these functions, *selectAttribute* is analogous to the "dot" operator of OO languages, which is used to access an object's attributes. For example, if *currentACEI* has a complex data type and one of its attributes is *critical_time*, then the value of *critical_time* can be extracted by the following GEL expression: "SelectAttribute("critical_time", currentACEI)".

3.2.2 GLIF3 constructs that return data consistent with the GEL data model

GLIF3's Get_Data_Action retrieves patient data from the EMR as HL7 RIM objects and transforms them to query result data types. It allows a mapping to be specified from GLIF3's default data model, the HL7 RIM, to GEL's data model. A guideline author can use Get_Data_Action to specify that an attribute of a complex RIM class is the source of data values for the query result, and that values of another attribute serve as the primary time in the query result. Thus, the query result is a list of value and primary time

[1] GEL currently does not support all of the numeric functions of Arden Syntax but they can be defined by GEL functions.

pairs similar to Arden's query result data type. However, the value attribute in GEL's query result holds a simple or a complex GEL type. Get_Data_Action specifies which data item from the EMR will serve as the source of data, and which attribute will be selected from the data item. In this way, specific attributes of the data item can serve as the source of the data, rather than the entire data item. For example, Get_Data_Action can retrieve all instances of Medication data items that refer to ACE Inhibitor treatments (Figure 2). It can assign the value of their *data value* attribute, which is a RIM Medication object (Figure 1), to the query result's "value" attribute, and assign the end time of each Medication treatment (*critical_time.high*) to the "primary time" attribute of the query result elements.

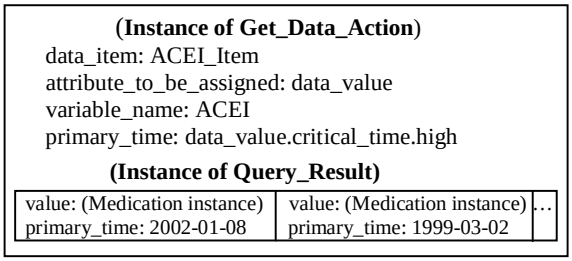

Figure 2. The Get_Data_Action and its query result that holds ACEI Medication objects data values and their primary times. In this example, the latest query result element has the primary_time 2002-01-08.

Because GEL query results have the same data model as Arden query results, many Arden operators can be used in GEL expressions. For example, GEL expressions can be written to select the latest element in an ACEI query result, to check that the latest prescription applies to the current date, or to form a list of current medications given to a patient out of single current medications. The Assignment_Action of GLIF3 can be used to set a variable with a result of a GEL expression. For example, a variable called current ACEI can be assigned with a GEL expression that returns the value of the "value" attribute of the **latest** ACEI element of the query result in Figure 2, provided its primary time is current (>= now), i.e., the medication prescription is still active. In the example shown in Figure 2, the primary time of the latest ACEI query result element is 2002-01-08. Figure 3(a) shows the GEL expression.

```
(a)  CurrentACEI := SelectAttribute("value", latest ACEI
     where time of it >= now)

(b) ACEIStartTime := SelectAttribute("low",
                     SelectAttribute("critical_time", currentACEI));
    ACEIStartTime < (now – 4 weeks)
```

Figure 3. GEL expressions

The variables that hold results of Get_Data and Assignment_Action can be used within decision and eligibility criteria expressed in GEL. Figure 3(b) specifies an expression and a criterion in GEL. The expression extracts the start time (critical_time.low) of the current ACE Inhibitor prescription from the value of variable currentACEI, extracted by the expression shown in Figure 3(a). The criterion checks whether an ACE Inhibitor was prescribed before 4 weeks ago.

4 Discussion

Our solutions can help guide others who are seeking to use Arden Syntax as an expression language that will be used with object-oriented EMR models. HL-7, within which both the RIM and Arden Syntax are being developed, is also seeking ways in which the two models can be harmonized. Our suggestions can provide possible direction to this effort.

The InterMed Collaboratory has explored the possibility of reconciling Arden Syntax's logical grammar with domain concepts and patient data represented as complex objects. GEL takes a conservative approach to reconciling the difference between Arden Syntax's data model and object-oriented medical data models. It does so by augmenting the data model in a limited way and adding several functions that enable (1) extraction of attributes from the complex data types, and (2) operations on lists of complex data values, while maintaining the existing Arden operators.

GEL's data model allows complex data types. However, GEL's expression language does not directly support the object-oriented model. Instead, the *selectAttribute* function of GEL is used to extract attribute values from complex types. In addition, user-defined functions can be used to convey the semantics of relationships provided by the OO model, such as the inheritance relationship discussed in Section 3.2.1. They can also be used to implement methods of object-oriented classes.

A disadvantage of GEL is its limitation for specifying expressions that relate to more than one of an object's attributes. For example, in a query result of AmoxicillinPrescriptions where the result values have a dose of 500 mg and an oral route, it is useful to write an expression that selects the start time from the values of the result. The "where" part of the expression would ideally be written as "where selectAttribute("route", it) = "oral"…". However, "it" cannot be an argument of a GEL function because the function needs to get the entire list or query result as a parameter. This restriction bars writing "where" clauses that use GEL functions. As a result, "where" clauses

cannot be used to check constraints on specific attribute values other than primary_time. Instead, there is a need to specify a loop that looks at single elements in the list or query result, and evaluates criteria imposed on the element's attribute values. Alternatively, specific GEL functions can be written to implement such loops for special cases.

We used GLIF3 for encoding two clinical guidelines: management of chronic cough and hypertension management. GEL was sufficient to encode the cough guidelines, but we had to add a special GEL function for encoding the hypertension guideline. While the special function allowed us to encode the second guideline, adding special-purpose functions is not a principled approach and there are likely to be additional expressions that cannot be encoded without adding more functions into GEL.

Other studies have tried to reconcile different data models. One study described a translation between complex data models using an intermediate OO rule-based approach.[8] When they translate an OO model into a non-OO model, the methods in the OO model cannot be translated, and they show up as a discrepancy between the two models. Another group developed a translation of high-order object queries to first-order relational queries.[9] We did not find published work that tried to reconcile an OO data model with an expression language that does not support operators on relations between attributes of complex schema/objects.

We are developing an OO expression language that can associate methods with each class in the OO medical data model. Binding methods with the data will provide a more systematic and scalable approach to defining an expression language. The OO expression language can have several predefined classes, such as List and Math, which contain Arden's list and numeric operators. For example, *merge* will be a method of the List class, whereas *sin* will be a method of the Math class. In addition to the built-in classes, the medical domain ontology classes will likely define methods to which the OO expression language can refer.

We have found that the Arden community also desires development of an expression language that can work with the object-oriented HL7 RIM data model. As part of the Arden and Clinical Guidelines special interest groups in HL7, we will be further developing the expression language that will work with the RIM. For the reasons we have outlined in this paper, we believe that designing an expression syntax that is designed from the outset to be compatible with the object-oriented data model is preferable to trying to reconcile two radically different data models.

Acknowledgements

Supported in part by Grant LM06594 with support from the Department of the Army, Agency for Healthcare Research and Quality, and the National Library of Medicine.

References

1. Sugden B, Purves IN, Booth N, Sowerby M. The PRODIGY Project - the Interactive Development of the Release One Model. AMIA Symp 1999; 359-363.

2. Fox J, Thomson R. Decision support and disease management: alogic engineering approach. *IEEE transactions on Information Tech in Biomed* 1998;2(4):1-12.

3. Musen MA, Tu SW, Das AK, Shahar Y. EON: A component-based approach to automation of protocol-directed therapy. *JAMIA* 1996;3:367-388.

4. Peleg M, Boxwala A, Ogunyemi O, et al. GLIF3: The Evolution of a Guideline Representation Format. Proc. AMIA Annual Symposium; 2000; p. 645-649.

5. Schadow G, Russler DC, Mead CN, McDonald CJ. Integrating Medical Information and Knowledge in the HL7 RIM. Proc. AMIA Annual Symposium 2000; p. 764-768.

6. Hripcsak G, Ludemann P, Pryor TA, Wigertz OB, Clayton PD. Rationale for the Arden Syntax. *Comput Biomed Res* 1994;27(4):291-324.

7. Jenders RA, Sujansky W, Broverman C, Chadwick M. Towards Improved Knowledge Sharing: Assessment of the HL7 Reference Information Model to Support Medical Logic Module Queries. Proc AMIA Annu Fall Symp; 1997; p. 308-312.

8. Su SYW, Fang SC, Lam H. An object-oriented rule-based approach to data model and schema translation. Florida: Departament of Computer and Information Sciences, University of Florida; 1992. Report No.: TR-92-015.

9. Quian X, Raschid L. Query Interoperation Among Object-Oriented and Relational Databases. In: Proc. of the 11th Intl. Conf. on Data Engineering; Taipei, Taiwan; 1995.