Guideline Interchange Format 3.0

Technical Specification

Draft Release

February 25, 2000

InterMed Collaboratory

Document Editors:

Mor Peleg, Aziz Boxwala

Significant contributions to the model and the document were made by (in alphabetical order):

Nachman Ash, Elmer Bernstam, Robert A. Greenes, Ronilda Lacson, Peter Mork, Omolola Ogunyemi, Lucila Ohno-Machado, Edward H. Shortliffe, Samson Tu, Qing Zeng

Table of Contents

1 Introduction 1
1.1 Purpose of this document 1
1.2 What is GLIF? 1
2 Status of document 1
3 Overview of GLIF3 2
3.1 Changes from version 2 2
3.1.1 Changes in the object model 2
3.1.2 Changes in the GLIF syntax 3
3.2 Guideline Abstraction Levels 3
3.3 Introduction to GLIF concepts 4
3.3.1 Guideline 4
3.3.2 Guideline steps 4
3.3.3 Complexity Management 6
3.3.4 Expressions 8
3.3.5 Data Model / Domain Ontology 9
3.3.6 Events and Exceptions 9
3.3.7 Supplemental Material 10
4 GLIF Logical model (UML class diagram) 11
5 GLIF Syntax (XML/RDF schema) 11
Bibliography 77
Appendix A Areas for future development 78
Appendix B Examples 78

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

1 Introduction

1.1 Purpose of this document

This document contains a technical specification of the Guideline Interchange Format (GLIF) version 3.0 (GLIF3). The specification includes a logical object-oriented model of guidelines and a text-based syntax for representing instances of guidelines. The document provides illustrations of various concepts for clarifying the meaning of the constructs that are specified. However, it is not intended to be a full tutorial on the use of GLIF.

1.2 What is GLIF?

GLIF is a specification for structured representation of guidelines. It was developed by the InterMed Collaboratory in order to facilitate sharing of clinical guidelines (Ohno-Machado, Gennari et al. 1998). The InterMed collaboratory was a joint project of medical informatics laboratories at Harvard (the Decision Systems Group at Brigham and Women's Hospital and Laboratory of Computer Science at Massachusetts General Hospital), Stanford, Columbia, and McGill Universities (Shortliffe, Barnett et al. 1996). That work is being continued under new funding by a subgroup of the InterMed collaborators, including the Decision Systems Group at Harvard, McGill, Columbia, Stanford, and the American College of Physicans-American Society of Internal Medicine.

The objective of the GLIF specification is to provide a representation for guidelines that have the following characteristics:

a. Precise

b. Non-ambiguous

c. "Human-readable"

d. Computable (in the sense that guidelines specified in GLIF may be used for computer-based decision support)

e. Independent of computing platforms (thus enabling sharing of guidelines)

Version 2.0 of GLIF (GLIF2) was published in 1998 (Ohno-Machado, Gennari et al. 1998). That version of GLIF has been the basis for several implementations of guideline-based applications, including one in the Brigham's BICS information system (Zielstorff, Teich et al. 1998). Web-based applications for driving clinical consultations (Boxwala, Greenes et al. 1999), and applications that search for eligible clinical protocols (Ohno-Machado, Wang et al. 1999).

2 Status of document

This document contains a draft of the GLIF3 specification that is still being developed. Readers should be aware that several aspects of the representation are not completely specified yet. Other parts of the representation may undergo changes before the final release of this version. A date for release of the final specification has not been decided on yet. However, we expect that date to be in the second quarter of 2000.

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

3 Overview of GLIF3

3.1 Changes from version 2

The changes in GLIF3 from GLIF2 are summarized. The GLIF2 specification document is available at http://www.glif.org/pubs/.

3.1.1 Changes in the object model

Representation in UML

The GLIF3 model is described via Unified Modeling Language (UML) class diagrams. Additional constraints on represented concepts are being specified in the Object Constraint Language (OCL), a part of the UML standard.

The GLIF2 specification was published in Interface Definition Language (IDL) (Object Management Group 1999).

Support for managing complexity in guideline instances

GLIF3 more fully defines a mechanism for specifying guideline steps iteratively through the nesting of subguidelines in steps. In addition to the action step, GLIF3 adds capability to specify decisions in an iterative manner.

GLIF3 will also include a capability for application-specific extensions in a scalable manner. These extensions can be built using the Macro_Step class. A macro can be used to represent application-specific constructs in a concise manner.

A capability to provide multiple views of the same guideline is also being added. This capability is provided through the use of filters that would collapse segments of the guideline from a default view of the guideline.

Expression specification

A structured grammar for specifying expressions and criteria is added to GLIF3. The grammar can be used for specifying logical criteria, numerical expressions, temporal expressions, and text string operations. The grammar has been derived from the Arden Syntax logic grammar (Clinical Decision Support & Arden Syntax Technical Committee of HL7 1999).

Domain ontology support

In order to facilitate use of standard medical vocabularies and integration of shared guidelines into clinical information systems environment, GLIF3 uses a layered approach for referencing clinical terms.

In GLIF2, definitions of patient data items were provided through a Patient Data class contained in an Action Specification. The Patient Data class is now obsolete.

Flexible decision model

GLIF3 provides a flexible decision model through a hierarchy of decision step classes. The hierarchy can be extended further in the future to support different decision models.

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

GLIF2 provided a Boolean decision model only.

Extended action specification model

The action specification model has been extended to include two different types of actions: guideline-flow relevant actions (such as calling a subguideline, computing values for data) and clinically relevant actions (recommendations). Clinically relevant actions reference the domain ontology.

A single action specification class was provided by GLIF2. While not explicitly described in the specification, structured descriptions of actions could be included by inheriting a subclass from this action specification class.

Other new concepts

Representation for several new concepts that did not exist in GLIF2 are added to GLIF3. These include specifications for

- Describing *Iterations* and conditions that control the iteration flow
- Describing *Events* and triggering of guideline steps by events.
- Describing *Exceptions* in guideline flow and associated exception-handling mechanisms.
- Representing *Patient State*.
- A *Keyword Didactic* for adding keywords to a variety of constructs in guidelines

Corrections to branch and synchronization step

The branch and synchronization step have been modified to remove redundancy in descriptions of parallel pathways in the guideline flowchart.

3.1.2 Changes in the GLIF syntax

XML-based syntax

The proprietary Object Data Interchange Format (ODIF)-based syntax (Pattison-Gordon 1996) in GLIF2 is being replaced with an XML-based syntax. A schema for the syntax is being developed.

3.2 Guideline Abstraction Levels

GLIF3 enables modeling guidelines in three levels of abstraction (Greenes TR Ref):

1. **Level A: abstract flowchart.** Guidelines that are specified in GLIF at this level can be used for browsing and reading in guideline viewing programs. However, these guidelines cannot be used for computation for decision support.
2. **Level B: computable level.** Guidelines specified at Level B may be validated for logical consistency and completeness (not correctness). For example, the syntax of expressions, definitions of patient data items and clinical actions, and flow of the algorithm would be specified at this level.
3. **Level C: implementable level.** At this level, guidelines are appropriate for

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

incorporation into institutional information system environments. Thus, these guidelines may have non-sharable concepts defined within them.

3.3 Introduction to GLIF concepts

In this section, we provide an overview of concepts specified in GLIF for representing instances of guidelines.

3.3.1 Guideline

A guideline is the root concept in an instance of a guideline. A guideline has attributes for defining the algorithm, guideline maintenance information, eligibility criteria, intention, didactics, exceptions for leaving the guideline, and the default viewer.

3.3.2 Guideline steps

The guideline steps are the nodes of the guideline's algorithm. The different types of guideline steps are: action, decision, branch, synchronization, patient_state and macro.

Action Step

Action steps specify clinical actions that are to be performed during the patient-care process. An action step specifies a set of tasks (Action_Specifications) that need to be performed. An action step has a single next-step. The action step has attributes that specify its iteration information, duration, triggering events, and associated exceptions. Action Steps are nested by including a Subguideline_Action type of task in the step. The Subguideline_Action task has a (sub)guideline attribute that contains the nested subguideline.

Action Specification

The Action_Specification class is used to describe a clinical action in detail. There are two types of Action Specifications: medically oriented and programming oriented. The programming-oriented action specifications include constructs for sending messages, generating events, accessing patient data, assigning values to variables, and calling a (sub) guideline. The medically-oriented action specification defines a medical_task, which is a concept defined in the Reference Information Model (RIM) layer of GLIF's data model/ontology. Examples of medically oriented action specifications include diagnostic test order, drug prescription, referral, and scheduling.

Branch Step

Branch steps are used to model concurrency. They direct flow to multiple guideline steps. All of these guideline steps may occur in parallel or in any order.

Synchronization Step

Synchronization steps are used in conjunction with branch steps. When a branch step is followed by multiple guideline steps, the flow of control must eventually converge in a single step. Each branch may lead to a series of steps, resulting in a set of branching paths. The step at which the paths converge is the synchronization step. When the flow of control reaches the synchronization step a continuation attribute specifies whether all,

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

some or one of the preceding steps must have been completed before control can move to the next step.

Patient State Step

A Patient State step is a guideline step that is used for two purposes. One is as a label that describes a patient state that is achieved by previous steps. This way, a guideline may be viewed as a state transition graph, where states are patient states, and transitions between these states are the networks of guideline steps (excluding patient state steps) that occur between two patient state steps. The other purpose of a patient state step is an entry point to the guideline. A patient state step has a criterion that describes the state of the patient who is at that patient state. A patient state step is followed by a single guideline step. It also has a label that defines the patient state in text that refers to terms taken from the domain ontology.

Decision Step

Decision Steps direct flow from one guideline step to another. There are two basic types of decision steps: **case** and **choice**. A decision step may link a guideline step to any other guideline step. A decision step contains a condition, which is an expression. The condition's value determines the control flow to one of a set of possible guideline steps, which are specified by the options of the decision step. The condition is compared, using an operator, to the different decision options. If the condition matches one of the decision options, then the control can flow, in the case of a choice step, and must flow, in the case of case step, to the next guideline step that is specified by that decision option. If the condition does not match any of the set of values specified by the decision options, or, if available data do not allow evaluation of the condition, then the control flows to the default next guideline step. Decision Steps are nested by specifying a (sub)guideline in the decision_detail attribute of the step. This subguideline is executed before the decision criterion for that step is evaluated. The subguideline would modify or create new variables and assign them values. The use of these variables in the decision criteria makes the decision nested. The action step has attributes that specify its iteration information, duration, triggering events, and associated exceptions.

Case Steps represent a deterministic decision between guideline steps. The different decision options are mutually exclusive. When a decision option condition value matches the case step's condition, then control flows to the destination specified by that decision option.

Choice Steps represent a non-deterministic decision between guideline steps. The different decision options are not necessarily mutually exclusive. Ranking the decision options depends on the class of choices. If the condition matches more than one of a set of values specified by the choices, a person will have to determine the next guideline step at run-time. Only one guideline step will be chosen to be the next step to be executed. Each option contains a degree of preference that may be modeled differently for the different types of choices (Rule-in, K-of-N, and Utility choices). The degree of preference will determine how the choices will be ranked. This will assist the user in choosing among the different options.

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Rule-in choices: a rule-in choice defines two criteria: rule-in and strict rule-in that specify what conditions may or must (respectively) hold for the choice to be applicable.

KofN choices: KofNChoices contain an array of criteria, each associated with a weight. The weighted criteria for each of the options will determine how an option will be ranked among the choices presented to a user at run-time. The sum of the weights for each criterion in a choice must equal 1.0. The higher the value of a choice (from 0 to 1), the higher its rank.

Utility Choice Steps and Utility Choices: utility choice steps represent a choice step that uses Utility theory in deciding among several options. It contains a pointer to the actual algorithm used to evaluate the choices. This may either be a decision analysis tree or an influence diagram. Utility choices contain the node that will determine its expected value or expected utility at run time. This expected utility would determine its rank among the list of choices.

3.3.3 Complexity Management

Guidelines can be very complex. A published guideline can span over a hundred pages of text. GLIF 3.0 introduces three mechanisms to manage the complexity of guidelines: nesting, views, and macros. Nesting enables the presentation of a guideline in a step-wise manner, working from a top-level specification, and gradually supplying finer levels of detail of action and decision steps. Figures 1 and 2 demonstrate how nesting is used to show the details of the treatment action step of the Stable Angina guideline (American College of Cardiology/American Hospital Association/American College of Physicians-American Society of Internal Medicine 1999). In addition to aiding in management of complex guidelines, nesting may also be useful in showing relationship to other guidelines by creating a top-level view that portrays the relationship. Since nesting allows grouping of parts of a guideline into a single unit, this is a mechanism that can allow model extensibility and reuse of part of a guideline as well as the adaptation of a guideline to a specific institution by replacement or elaboration of specifications for well-defined sections of a guideline (e.g., replacing a goal with a procedure).

Figure 1. The details of the Treatment algorithm, which is part of the Stable Angina Guideline

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Figure 2. The details of the Education and Risk Factor Modification action step, shown in Figure 1. Given the fact that aspirin clearly reduces the risk of subsequent heart attack and death but has no known benefit in reducing angina, the authors assigned it to the education and risk factor component

Views control the amount of detail displayed by default for each viewer. For example, a radiation oncologist looking at a breast cancer guideline will see (by default) relatively little detail about screening, diagnosis and surgery. He will see a great deal of detail regarding radiation therapy, however. Although we anticipate that the most common use of views will be to support different kinds of users, there may be other relevant view dimensions (e.g. situation such as routine vs. disaster).

A macro step represents a pattern that often appears in guidelines and is equivalent to a

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

sequence of non-macro GLIF steps. For example, a clinical trial protocol follows a pattern of phases, where each phase consists of visits. During a visit, the trial coordinator usually performs certain actions (orders tests, delivers treatment) and completes case report forms. Using macro steps to represent commonly occurring patterns has benefits for authoring, visual understanding, and execution of guidelines. Macros deal with domain-level concepts (e.g., visits, forms) rather than action, branch, or decision steps. A macro step is a (potentially, though not necessarily, nested) guideline step that describes such patterns of GLIF components in a custom-made step. The attributes of the macro step are the parameters of the pattern of GLIF components. For example, the attributes of a *Clinical Trial Visit* macro step are the time of the visit, the tasks to be performed during the visit, the forms to be completed, and decision steps. The GLIF pattern of a Macro Step is described in the Macro Schema object that is referenced by the macro step. This pattern maps attributes of the macro to components in a GLIF step. Thus, using the schema and the values of attributes, a complete sequence of GLIF objects can be constructed. The schema is also used for validation of macro attributes. Because the knowledge of the macro can be transformed to a sequence of GLIF components, even applications that are unaware of particular macros and their attributes can display and execute the underlying GLIF. In this way, macros can be used to create application-specific components with minimal effect on sharability of GLIF. A notation/language for GLIF patterns still has to be developed. This language will be used to describe schemas abstractly. The definition of this language will require significant effort, which we will defer until other aspects of GLIF are more fully developed. We are considering a notation similar to the XML schema definition syntax for the World Wide Web Consortium (W3C).

3.3.4 Expressions

GLIF supports specification of different types of expressions: criteria, temporal expression, list expression, numerical expression and string expression. Different expression languages can be used. Currently, BNF grammar is given for a modification[1] of the Arden Syntax logic grammar, called GLIF_ARDEN. While this gives guideline users the flexibility to specify criteria in a language of their choosing, it is possible that the resulting guidelines will not be sharable because constructs supported by one language (e.g., temporal constructs) cannot be expressed in another language. Two types of criteria are possible: three-valued, and probabilistic. Three-valued criteria evaluate to true, false, or unknown. Probabilistic criteria are used to add conditional probabilities or Bayesian network information.

[1] A modification of the Arden syntax is used, and not Arden syntax itself, since we wanted to be able to express certain operators, which are not supported by Arden syntax, but are needed for specifying criteria. These operators are “is a”, “overlaps”, “xor”, “from now”, “is unknown” and “at least *k* of…”. GLIF made minor changes to the syntax of some operators to allow a more natural way of expressing them; the operator “occur/occurs/occurred at” has the same meaning as “occur/occurs/occurred equals” and the operators “in”, “not in” has the same meaning as Arden’s “is in”, “is not in”. All of these operators, including the original Arden syntax format, can be used. All of Arden’s logic grammar is supported by GLIF.

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women’s Hospital.

3.3.5 Data Model / Domain Ontology

GLIF 3.0's data model / domain ontology is organized into three layers, which are: core GLIF, GLIF Reference Information Model (RIM), and GLIF's medical ontology.

Core GLIF

Core GLIF is a part of the GLIF standard that deals with data modeling issues. It defines how medical data and concepts may be represented and referenced by GLIF. It also defines the scope of data items and how the data items acquire their values.

GLIF Reference Information Model (RIM)

The RIM provides a semantic hierarchy for medical concepts that represent different classes of medical data. The hierarchy allows specification of the attributes of each class of medical data in an object-oriented fashion. It also makes it possible to perform type checking. Unlike core GLIF, users have an option to use the GLIF RIM or not. However, unless different users are using the same RIM, it will be very hard for them to share with each other any reference to patient data in a structure way.

Medical Ontology

The medical ontology for GLIF should contain a term dictionary and a medical knowledge base. Information on how to code data is beyond the scope of a guideline format, so a user can choose to use any ontology with GLIF. However, without sharing a common term dictionary, users may not be able to share the meaning of the criteria and actions in structured way. Therefore, it is very important for users to have a common term dictionary and we found UMLS is very appropriate for that purpose. The more medical knowledge we include in GLIF, the better we can assist users in guideline authoring and execution. To acquire and represent medical knowledge is a huge research area by itself. What we want to achieve is to study what types of knowledge are needed and demonstrate how GLIF 3.0 can incorporate them.

3.3.6 Events and Exceptions

Action steps and decision steps have an attribute, called triggering_events, which specifies what events trigger the start of the step, and the earliest and latest times after which the step should be started following the occurrence of the triggering event. When a guideline step finished its execution and the control flow is about to pass to the next step, then, if the next step has associated triggering events, then this next step is executed only after one of its triggering events occurred. Any of the triggering events that occur can trigger the step that is pending. In cases where several triggering events are defined (each with its own start time constraints), their priorities are compared, and the highest priority event is chosen to trigger the step. Different Event types are possible. They include: end of a previous guideline step, patient arrival, patient data availability, and temporal events, such as that a certain point of time has arrived.

Action steps and decision steps and the guideline class have an attribute, called exceptions, which specifies an exception-event that should be checked for during

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

execution, a (guarding) condition, and a next step. If the exception event occurs and the condition holds, then the step associated with the exception is terminated, and control passes to the next step that is defined by the exception. The condition is part of the exception (and not a separate decision step which is specified as the exception's next_step) so as not to unnecessarily leave a step, stopping its execution, when a triggering_event occurs, unless the condition holds. Any of the exceptions that occur can stop the execution of the current step and pass control to another step. In cases where several exceptions are defined (each with its own next_step), their priorities are compared, and the highest priority exception is chosen to trigger the step.

3.3.7 Supplemental Material

The Supplemental Material class provides background or supporting information for Guidelines and guideline steps. There are three different types of supplemental material classes: Local_Material that points to locally stored material, WWW_Material that references a URL, and Keyword_Didactic that is used to reference keywords that can be used for search purposes.

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

4 GLIF Logical model (UML class diagram)

GLIF

Thu Feb 17 11:30:05 PST 2000

Subpackages

package Action_Specification_Package

package Data

package EventsAndExCeptions

package Expression_Package

package Guideline_Package

package Guideline_Step_Package

package Iteration_Package

package Nesting

package Supplemental_Material_Package

package Temp

package Temporal_Expression_Package

package Views_Package

Classes

class Guideline_Model_Entity

Class Diagram

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

5 Package Node Detail

Package *Action_Specification_Package*

Package *Data*

Description:
Decision steps direct flow from one *guideline step* to another. A *decision step* may link a *guideline step* to any other *guideline step*. A *decision step* contains a condition, which is an expression. The *condition's* value determines the control flow to one of a set of possible *guideline steps*, which are specified by the *options* of the *decision step*. The *condition* is compared, using an *operator* to the *options.condition_value*. If the *condition* matches one of the *decision options* then the control can flow, in the case of a choice step, and must flow, in the case of case step, to the *guideline step* that is specified by that *decision option's destination*. If the *condition* does not match any of the set of values specified by the *decision options*, or, if available data do not allow evaluation of the *condition*, then the control flows to the default destination *guideline step*.

When a guideline step finished its execution and the control flow is about to pass to the next step, then, if the next step has associated triggering events, then this next step is executed only after one of its triggering event occurred.

Purpose:
The *conditional step* used an extended Boolean model. This made it cumbersome and error-prone to represent criteria that do not have a true-or-false outcome (e.g., what is the patient's age category: neonate, infant, toddler, child, adolescence, adult, elderly). Therefore, the *case step* replaced the *conditional step* by allowing a conditional choice to be made among several alternative *guideline steps*.

Considerations:
1.The operator was added so that different comparisons would be possible. For example the operator may be "=", or may be "in" to represent a comparison to an interval (age in (30,40) as opposed to age eq > 30)

Package *EventsAndExCeptions*

Different Event types are defined: end of a previous guideline step, patient arrival, patient data availability, and temporal events, such as a certain point of time has arrived.

Action- and decision steps have an attribute, called triggering_events, which specifies what events trigger the start of the step, and with what temporal constraints. The triggering events are of the class EventStartTimeConstraint. This class specifies an event and the earliest and latest times after which the step should be started following the occurrence of the triggering event.

Action- and decision steps have an attribute, called exceptions, which specifies what exceptions should be checked during the execution of the step. The exceptions are of the class Exception. This class specifies the exception-event that should be checked for, a (guarding) condition and a next step. If the exception event occurs and the condition holds, then we terminate the step associated with the exception, and move on to the next step that is defined by the exception.

Package *Expression_Package*

The Expression class is a parent Class for all expressions: criterion, temporal expression, list expression, numerical expression and string expression. Different expression languages can be used. Currently, BNF grammar is given for a modification of the Arden Syntax logic grammar, called GLIF_ARDEN. While this gives guideline users the flexibility to specify criteria in a language of their choosing, it is possible that the resulting guidelines will not be sharable because constructs supported by one language (e.g., temporal constructs) cannot be expressed in another language.

Package *Guideline_Package*

Package *Guideline_Step_Package*

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Dependency Links

to **ClassDiagram** Action_Specification_Package

automatic

to **ClassDiagram** Supplemental_Material_Package

automatic

Package *Iteration_Package*

The Iteration_Specification class specifies information regarding the loop structure of the iteration. The action- and decision steps that should be iterated, that references the Iteration_Specification, are iterated until the abort condition or stopping condition criteria hold. The iterations are carried out at a certain (fuzzy) frequency (e.g., every 8h +/- 30min or, 3 times a day). If the iteration does occur within the specified frequency, then the iteration time points remain according to schedule, and are not reset (e.g., the patient came in after 8 1/2 hours. The next iteration should take place after 7 1/2 hours). Some frequencies may have a specified allowed offset. This means that if the iteration began outside the fuzzy frequency, but within the offset, then the iteration takes place and the iteration points are reset. For example, the patient came back after 9 hours. He gets treated (since the offset is 8 +/-2 h) but the next iteration is 8 hours late.

Package *Nesting*

Nesting

Nesting of guideline steps is supported in the folowing manner:

Action Steps are nested by including a Subguideline_Action type of task in the step. The Subguideline_Action task has a subguideline attribute that contains the nested subguideline.

Decision Steps are nested by specifying a subguideline in the decision_detail attribute of the step. This subguideline is executed before the decision criterion for that step is evaluatedThe subguideline would modify or create new variables and assign them values. The use of these variables in the decision criteria makes the decision nested.

A nested action action or decision can potentially contain any kind of step including other actions, decisions, and other nested steps. Nesting is very useful for managing the complexity of guidelines. Nesting enables looking at a guideline from a top-level view and then zooming into/out of some of its parts. Nesting is also useful in representing a guideline in the context of other guidelines. Since nesting allows grouping of parts of a guideline into a single unit, this is a mechanism that can allow model extensibility and reuse of part of a guideline (defining macros), or adaptation of a guideline to a specific institution by replacing specifications for parts of a guideline (i.e., replacing a goal with a procedure). By use of nesting, steps will be defined as containing other steps. As such, the guideline will be iteratively refined from a coarse level to a fine level of detail

Package *Supplemental_Material_Package*

Package *Temporal_Expression_Package*

Package *Views_Package*

Class Detail

Class *Guideline_Model_Entity*

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Attributes

name

Attribute Detail

name

Package *Action_Specification_Package*

Classes

class Action_Specification_Package.Action_Specification

class Action_Specification_Package.Assignment_Action

class Action_Specification_Package.Generate_Event_Action

class Action_Specification_Package.Get_Data_Action

class Action_Specification_Package.Medically_Oriented_Action_Specification

class Action_Specification_Package.Message_Action

class Action_Specification_Package.Programming_Oriented_Action_Specification

class Action_Specification_Package.Subguideline_Action

Class Diagram

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Class Detail

Class *Action_Specification_Package.Action_Specification*

Inherits from:

Guideline_Model_Entity

Description:
The *Action_Specification* class is used to describe a clinical action in detail. This is an abstract base class. We expect that there will be a hierarchy of subclasses that describe different types of actions in a structured manner.

Purpose:
To specify details of an action.
Considerations:
This class was derived from the GLIF-2 Action_Spec class. The name was changed for consistency.

Attributes

description

intention

Attribute Detail

description

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Data type: String
Multiplicity: 0:1
Description: A brief human-readable description of the action to be performed.
Level: A

intention

Class *Action_Specification_Package.Assignment_Action*

Inherits from:

Action_Specification_Package.Programming_Oriented_Action_Specification

Description:
The *Assignment_Action* class is used to create/instantiate a new data item. This data item is assigned the value resulting from the evaluation of the expression.

Purpose:
To instantiate a new data item that is assigned the result from the expression evaluation.
Considerations:

- This class may need some modifications once the patient data items are better defined. The new data item that is created may need more definition in terms of its datatype or relationships with other data items.
- The datatype of data_item_name needs to be restricted to a type that allows only names which are legal (starts with letter, no +,-, etc.)

Attributes

data_item

expression

Attribute Detail

data_item

Data type: Patient_Data
Multiplicity: 0:1
Description: The definition of the data that is output by this action.
Level: A

expression

Data type: Expression
Multiplicity: 0:1
Description: The expression to be evaluated. If expression is null, a new data item with null value is created
Level: B

Class *Action_Specification_Package.Generate_Event_Action*

Inherits from:

Action_Specification_Package.Programming_Oriented_Action_Specification

Attributes

event

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Attribute Detail

event

Class *Action_Specification_Package.Get_Data_Action*

Inherits from:

Action_Specification_Package.Programming_Oriented_Action_Specification

Attributes

data_item

Attribute Detail

data_item

Data type: Variable_Data_Item
Multiplicity: 0:1
Description: The data that is obtained by this action.
Level: A

Class *Action_Specification_Package.Medically_Oriented_Action_Specification*

Inherits from:

Action_Specification_Package.Action_Specification

MedicallyOrientedActionSpecification defines a medical_task, which is a concept defined in the RIM layer of GLIF's data model/ontology. Examples of MedicallyOrientedActionSpecification include Diagnostic_Test_Order, Drug_Prescription_Action, Refferal, and Scheduling.

Attributes

medical_task

Attribute Detail

medical_task

Class *Action_Specification_Package.Message_Action*

Inherits from:

Action_Specification_Package.Programming_Oriented_Action_Specification

Description:
The *Message_Action* class is used to send a message to a destination

Purpose:
similar to the "write" statement of Arden Syntax.
Considerations:

Attributes

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

destination

message

Attribute Detail

destination

Data type: String
Multiplicity: 0:1
Description: The message' destination
Level: C

message

Data type: String
Multiplicity: 1:1
Description: The message to be sent. the string can include text and variables (e.g., "patient age is: " || age || "\n"
Currently, the specification does not provide any formatting commands.
Level: C

Class *Action_Specification_Package.Programming_Oriented_Action_Specification*

Inherits from:

Action_Specification_Package.Action_Specification

Class *Action_Specification_Package.Subguideline_Action*

Inherits from:

Action_Specification_Package.Programming_Oriented_Action_Specification

Description:
The *Subguideline_Action* class is a specialization of the *Action_Specification* class. An object of this class when contained inside an action_step creates a nested action step.

Purpose:
The Subguideline_Action is used to describe nested actions.

Considerations:
A design consideration was to include a subguideline attribute in the Action_Step class rather than in an Action_Specification (as in GLIF-2). However, that option was not semantically clear in terms of the relationships amongst the subguideline and the other actions that are included in the Action_Step. the other actions that get inc

Attributes

action_detail

Attribute Detail

action_detail

Data type: Subguideline
Multiplicity: 0:1
Description: The nested subguideline that is called by the action specification.
Level: B

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Package *Data*

Classes

class Data.Data_Item

class Data.Literal_Data_Item

class Data.Data_Item_List

class Data.Variable_Data_Item

Class Diagram

Description:
Decision steps direct flow from one *guideline step* to another. A *decision step* may link a *guideline step* to any other *guideline step*. A *decision step* contains a condition, which is an expression. The *condition's* value determines the control flow to one of a set of possible *guideline steps*, which are specified by the *options* of the *decision step*. The *condition* is compared, using an *operator* to the *options.condition_value*. If the *condition* matches one of the *decision options* then the control can flow, in the case of a choice step, and must flow, in the case of case step, to the *guideline step* that is specified by that *decision option's destination*. If the *condition* does not match any of the set of values specified by the *decision options*, or, if available data do not allow evaluation of the *condition*, then the control flows to the default destination *guideline step*.

When a guideline step finished its execution and the control flow is about to pass to the next step, then, if the next step has associated triggering events, then this next step is executed only after one of its triggering event occurred.

Purpose:
The *conditional step* used an extended Boolean model. This made it cumbersome and error-prone to represent criteria that do not have a true-or-false outcome (e.g., what is the patient's age category: neonate, infant, toddler, child, adolescence, adult, elderly). Therefore, the *case step* replaced the *conditional step* by allowing a conditional choice to be made among several alternative

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

guideline steps.

Considerations:
1.The operator was added so that different comparisons would be possible. For example the operator may be "=", or may be "in" to represent a comparison to an interval (age in (30,40) as opposed to age eq > 30)

Class Detail

Class *Data.Data_Item*

Inherits from:

Guideline_Model_Entity

Description:
Data Item is a symbol which stores value that may be patient data reffered to by the guideline or data computed or created by guidelines.

Purpose:
Data items are needed for the expression in *decision step* and the action spec in *action step*. Operands in expressions can be data items and data items can be created/instancianted in *get and assignment statements*.

Considerations:
Data item is a supr class for *variable data item* and *literal data item*. This class was created to capture the common attributes shared by the *variable data item* and *literal data item*.

Attributes

name

ontology

referring_concept

RIM

Attribute Detail

name

Data type: String
Multiplicity: 1
description: Name is what users want to refer to a data item. Referring to the same concept, users may choose different names for a data item. For example, in medical record, 'sex' and 'gender' can refer to the same concept. Also, sometimes, a user may want to use a abbreviation instead of the whole concept name.
Level: A, B, C

ontology

Data type: String
Multiplicity: 1
description: Ontology indicates what onotlogy the concepts which are referred to by the data item belong to. When no Ontology is used or no map can be found for the name of the data item, this attribute will take the default value of "unknown". Otherwise the value of the attribute will be the identifier of the ontology.
Level: B, C

referring_concept

Data type: ConceptCode
Multiplicity: 1
description: Referring concept indicates what concept the data item refers to. Each data item should refer to some concept. The concept can represent simple data types such as integer or it can be more complex types like 'medication order'. The concepts will

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

be referred to by their codes. The term dictionary in the medical ontology will determine the code of a concept.One data item can only refer to one concept, but multiple data items may refer to the same concept. When no Ontology is used or no map can be found for the name of the data item, this attribute will take the default value of "unknown". Otherwise the value of the attribute will be the identifier of the cocnept.
Level: B, C

RIM

Data type: String
Multiplicity: 1
description: RIM indicates what data model the data item will use for value representation. When no RIM is used, this attribute will take the default value of "unknown". Otherwise the value of the attribute will be the identifier of the RIM model.
Level: B, C

Class *Data.Data_Item_List*

Inherits from:

Guideline_Model_Entity

Description:
Data Item List is a list of data items used by the guideline.

Purpose:
Data item list is used to store all the data items in a guideline. Putting all data items in list makes it easier to access and update data items.

Considerations:
Each *data item* need to be stored in the guideline. Having a list instead of placing the *data items* before where they are used

Attributes

entry_array

Attribute Detail

entry_array

Data type: Array of data items
Multiplicity: 1
description: this is a simple array and the elements in the array are data items.
Level: B, C

Class *Data.Literal_Data_Item*

Inherits from:

Data.Data_Item

Description:
Literal data Item is a *data Item* with fixed values.

Purpose:
Literal data item is one way to refer to a absract concept object with no specific instance. Although this does not imply the concept can not have any instance, the *Literal data item* that refers to the concept will not have any data value. It can also be used represent a specific instance of a concept.

Considerations:
.

Attributes

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

data_value

Attribute Detail

data_value

Data type: Any (Data type must be the type specified by the referring_concept)
Multiplicity: 0...n
description: Data value stores the value of the literal data item. Data value of a data item will be modeled by zero or one instance of data model objects in the RIM model. The data model object of the instance is that of the concept the data item refers to. No all literal data items has instance. In some cases, a literal represents an instance of a concept instead of the concept itself. So, in such cases, a literal can have a data value of one instance. For example, in 'height > 5 foot 4 inch' 5 foot 4 inch refers to a concept 'length', but it is an instance of the concept instead of the concept itself. When no value is specified, this attribute will be an empty list.
Level: B, C

Class *Data.Variable_Data_Item*

Inherits from:

Data.Data_Item

Description:
Variable data Item is a *data Item* with mutable values.

Purpose:
Variable data item is used to store patient data such as lab test results for calculation or evaluation. It can also be used store data such 'risk' which are created and computed by the guideline.

Considerations:
The same symbol may be used through a guideline and its value may change as a result of retrieval of new data and new assigment.

Attributes

data_value

owner

Attribute Detail

data_value

Data type: Any (Data type must be the type specified by the referring_concept)
Multiplicity: 0...n
description: Data value stores the value of the variable data item. Data value of a variable data item will be modeled by a list of instances of data model objects in the RIM model. The data model object of the instance is that of the concept the data item refers to. For example, a variable data item 'albumin' may refer to 'Serum Albumin Test Result'. So the data value is list of instances of 'Serum Albumin Test Result' object in the RIM. When no value is specified, this attribute will be an empty list.
Level: B, C

owner

Data type: String
Multiplicity: 1
description: Owner indicates who the patient is when patient data or other data specific to a particular patient. The reason to specify this is because sometimes even in one guideline, data from multiple patient will be mentioned although most guideline we have seen so far are specific for one patient. For example, clinical trial guidelines sometimes refer to a group of patients. For individual patients, we may be able to use some kind of MRN to distinguish one patient from another. For groups of patient, since we may not know or need to know each patient's identity, we propose to use free text to describe such groups. When no owner is specified, this attribute will take the default value of "unknown". Otherwise the value of the attribute will be the identifier of the owner.
Level: B, C

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Package *EventsAndExCeptions*

Classes

class EventsAndExCeptions.Event

class EventsAndExCeptions.Event_Start_TimeConstraint

class EventsAndExCeptions.Exception

Interfaces

interface EventsAndExCeptions.Event_Type

Class Diagram

Different Event types are defined: end of a previous guideline step, patient arrival, patient data availability, and temporal events, such as a certain point of time has arrived.

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Action- and decision steps have an attribute, called triggering_events, which specifies what events trigger the start of the step, and with what temporal constraints. The triggering events are of the class EventStartTimeConstraint. This class specifies an event and the earliest and latest times after which the step should be started following the occurrence of the triggering event.

Action- and decision steps have an attribute, called exceptions, which specifies what exceptions should be checked during the execution of the step. The exceptions are of the class Exception. This class specifies the exception-event that should be checked for, a (guarding) condition and a next step. If the exception event occurs and the condition holds, then we terminate the step associated with the exception, and move on to the next step that is defined by the exception.

Interface Node Detail

Interface *EventsAndExCeptions.Event_Type*

More event types may be added.

Class Detail

Class *EventsAndExCeptions.Event*

Inherits from:

Guideline_Model_Entity

Description: This class contains a triggering event and the earliest and latest times after which a system response to the event should be generated. The event triggers action or decision steps within the latest_start_time and not before the earliest_start_time.

Considerations
Currently, complex events are not modeled. To express an event such as "1 hour after event E1", we use the triggering event as E1 and an earliest start time of 1 hour. Complex events such as "DowJones changed by 20% in any 2 hour interval" are not possible to model right now.

Attributes

event_name

event_type

Attribute Detail

event_name

event_type

Class *EventsAndExCeptions.Event_Start_TimeConstraint*

Inherits from:

Guideline_Model_Entity

Description: This class contains a triggering event and the earliest and latest times after which a system response to the event should be generated. The event triggers action or decision steps within the latest_start_time and not before the earliest_start_time.

Any of the triggering events that occur can trigger the step that is about to be triggered. In cases where several triggering events are defined (each with its own start time constraints), their priorities are compared, and the highest priority event is chosen to trigger the step.

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Attributes

earliest_start_time

latest_start_time

priority

triggering_event

Attribute Detail

earliest_start_time

Data type: Time_Literal_Expression
Multiplicity: 1
Description:the earliest after which a system response should be generated after the triggering event occurred
Level: B, C

latest_start_time

Data type: Time_Literal_Expression
Multiplicity: 1
Description:the latest after which a system response should be generated after the triggering event occurred
Level: B, C

priority

Data type: integer
Multiplicity: 0:1
Description: the priority of the exception
Level: B and C

triggering_event

Data type: Event
Multiplicity: 1
Description:the event
Level: B, C

Class *EventsAndExCeptions.Exception*

Inherits from:

Guideline_Model_Entity

Action- and decision steps and the guideline class have an attribute, called exceptions, which specifies what exceptions should be checked during the execution of the step. The exceptions are of the class Exception. This class specifies the exception-event that should be checked for, a (guarding) condition and a next step. If the exception event occurs and the condition holds, then we terminate the step associated with the exception, and move on to the next step that is defined by the exception. The condition is part of the exception (and not a separate decision step which is specified as the exception's next_step) since we do not want to unnecessarily leave a step, stopping its execution, when a triggering_event occurs, unless the condition holds.

Any of the exceptions that occur can stop the execution of the current step and pass control to another step. In cases where several exceptions are defined (each with its own next_step), their priorities are compared, and the highest priority exception is chosen to trigger the step.

Attributes

condition

exception_event

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Interfaces

interface Expression_Package.Expression_Language

interface Expression_Package.Three_Valued_Type

Class Diagram

The Expression class is a parent Class for all expressions: criterion, temporal expression, list expression, numerical expression and string expression. Different expression languages can be used. Currently, BNF grammar is given for a modification of the Arden Syntax logic grammar, called GLIF_ARDEN. While this gives guideline users the flexibility to specify criteria in a language of their choosing, it is possible that the resulting guidelines will not be sharable because constructs supported by one language (e.g., temporal constructs) cannot be expressed in another language.

Interface Node Detail

- **Interface** *Expression_Package.Expression_Language*

- **Interface** *Expression_Package.Three_Valued_Type*

Class Detail

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Class *Expression_Package.Criterion*

Inherits from:

Expression_Package.Expression

Criteria: There are two types of criteria: three-valued, and probabilistic. Three-valued criteria evaluate to true, false, or unknown. Probabilistic criteria are used to add conditional probabilities or Bayesian network information. With probabilistic criteria, the dependence probabilities among all variables in a specification have to be known in order for the criterion to be evaluated. The problem is that the number of conditional probabilities that need to be specified increases exponentially as the number of variables used in a specification increase, and the dependence relationships among some variables may not be known. Users of probabilistic criteria need to provide the conditional probabilities needed for evaluating these criteria, and this means that use will probably be limited to advanced users of GLIF.

Class *Expression_Package.Expression*

Inherits from:

Guideline_Model_Entity

Description:

Purpose

Parent Class for all expressions: logical expressions (criteria), temporal expressions, arithmetic expressions, literals, and functions

Considerations:
There is no class Two-Valued_Criterion because expressions involving boolean/two-valued criteria can be handled using three-valued criteria.
The attribute, Expression.encoding_language was added to provide a means of using different expression languages (besides GLIF_ARDEN) in GLIF. While this gives guideline users the flexibility to specify criteria in a language of their choosing, it is possible that the resulting guidelines will not be sharable because constructs supported by one language (e.g., temporal constructs) cannot be expressed in another language. While flexibility is an admirable goal, more consideration should be given to the long-term consequences of allowing different expression languages if the goal of making guidelines sharable is to be achieved.
With probabilistic criteria, the dependence probabilities among all variables in a specification have to be known in order for the criterion to be evaluated. The problem is that the number of conditional probabilities that need to be specified increases exponentially as the number of variables used in a specification increase, and the dependence relationships among some variables may not be known. Users of probabilistic criteria need to provide the conditional probabilities needed for evaluating these criteria, and this means that use will probably be limited to **advanced** users of GLIF.
Evaluation of the structured string in an expression specification is performed within the parser for the grammar corresponding to Expression.encoding_language. Evaluation occurs after the entire string has been parsed. A table/database that stores variable names and values is needed in order to perform the evaluations. For probabilistic criteria, this table/database would have to include the conditional probabilities needed for evaluating the expression.
Partial specification of GLIF_ARDEN grammar in Backus Naur Form (BNF): (The grammar has been expanded to include almost all of the operators that Arden Syntax allows – 01/06/2000) A modification of the Arden syntax is used, and not Arden syntax itself, since we wanted to be able to express certain operators, which are not supported by Arden syntax, but are needed for specifying criteria. These operators are "is a", "overlaps", "xor", "from now", "is unknown" and "at least k of ...". GLIF made minor changes to the syntax of some operators to allow a more natural way of expressing them; the operator "occur/occurs/occurred at" has the same meaning as "occur/occurs/occurred equals" and the operators "in", "not in" has the same meaning as Arden's "is in", "is not in". All of these operators, including the original Arden syntax format, can be used. All of Arden's logic grammar is supported by GLIF.

Expression: LogicalExpression | Function | WhereExpression | TemporalExpression | (Expression) | Literal | NumericalExpression | StringExpression | ListExpression

LogicalExpression: UnaryOperator Expression | Expression UnaryOperator | Expression BinaryOperator Expression | at least Number of LogicalExprList | UnaryOperator LogicalExprList | TernaryOpExpression

Function: Identifier (ExpressionList)

WhereExpression : DEFINITION TO COME

TemporalExpression: TimeLiteral | Duration | TimeInterval | Frequency | FrequencySet | Duration UnaryOperator | UnaryOperator Number of Identifier | UnaryOperator Identifier | TemporalExpression BinaryOperator TemporalExpression | Identifier BinaryOperator TemporalExpression | TemporalExpression BinaryOperator Identifier | TimeInterval lasting Duration to Duration

Literal: Identifier | Number | StringLiteral | Tri-StateValue | ListLiteral | TimeLiteral

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

NumericalExpression: Number | UnaryOperator Expression | Expression BinaryOperator Expression

StringExpression: DEFINITION TO COME

ListExpression: FULL DEFINITION TO COME (since GLIF will support lists) Definitions will include: last (Number, etc) from (ListLiteral or Identifier) | first (Number, etc) from (ListLiteral or Identifier) | latest (Number, etc) from (ListLiteral or Identifier) | earliest (Number, etc) from (ListLiteral or Identifier) | nearest (Number, etc) from (ListLiteral or Identifier)

ExpressionList: -nothing- | Expression | Expression , ExpressionList

LogicalExprList: LogicalExpression | LogicalExpression, LogicalExprList

Duration: Number UnaryOperator

Frequency: EveryFrequency | TimesFrequency | Frequency for Duration

EveryFrequency: every Duration | every Duration with window -Duration, +Duration, offset -Duration, +Duration

TimesFrequency: Number times a TimeUnit | Number times | Number times per Duration

FrequencySet: Frequency followed by FrequencySet | Frequency

TimeUnit: years | year | months | month | weeks | week | days | day | hours | hour | minutes | minute | seconds | second

TimeInterval: TimeLiteral to TimeLiteral | Identifier to Identifier

UnaryOperator: not | ! | - | ago | from now | IsAreWasWere null | IsAreWasWere not present | IsAreWasWere not null | IsAreWasWere present | IsAreWasWere boolean | IsAreWasWere not boolean | IsAreWasWere unknown | IsAreWasWere number | IsAreWasWere not number | IsAreWasWere time | IsAreWasWere not time | IsAreWasWere duration | IsAreWasWere not duration | IsAreWasWere string| IsAreWasWere not string | IsAreWasWere list| IsAreWasWere not list | years | year | year of | months | month | month of | weeks | week | week of | days | day | day of | hours | hour | hour of | minutes | minute | minute of | seconds | second |second of | exists | exist | exists of | exist of | any of | all of | last | last of | first | first of | latest | latest of |earliest | earliest of | time of |extract month

BinaryOperator: , | or | | | xor | *| | & | and | == | = | eq | IsAreWasWere equal | != | <> | ne | IsAreWasWere not equal | < | lt | IsAreWasWere less than | IsAreWasWere not greater than or equal | <= | le | IsAreWasWere less than or equal | IsAreWasWere not greater than | > | gt | IsAreWasWere greater than | IsAreWasWere not less than or equal | >= | ge | IsAreWasWere greater than or equal | IsAreWasWere not less than | IsAreWasWere before | IsAreWasWere not before | IsAreWasWere after | IsAreWasWere not after | OccurOccursOccurred at | OccurOccursOccurred equal | OccurOccursOccurred before | OccurOccursOccurred not before | OccurOccursOccurred after | OccurOccursOccurred not after | OccurOccursOccurred at | OccurOccursOccurred equal | in | IsAreWasWere in | not in | IsAreWasWere not in | + | - | * | / | | ** | ^ | is a | is-a | before | after | | matches pattern

IsAreWasWere: is | are | was | were

OccurOccursOccurred: occur | occurs | occurred

TernaryOpExpression: Expression IsAreWasWere within Expression EndTernExpr | Expression IsAreWasWere not within Expression EndTernExpr | Expression OccurOccursOccurred within Expression EndTernExpr | Expression OccurOccursOccurred not within Expression EndTernExpr

EndTernExpr: to Expression | preceding Expression | following Expression | surrounding Expression

Identifier: [A-Za-z0-9_]+

Number: [0-9]+.?[0-9]* UnitSpec

StringLiteral: "[^"]*"

Tri-StateValue: true | false | unknown

ListLiteral: [ExpressionList]

TimeLiteral: year-month-dayThours:minutes:seconds.fraction(Z|+/-hours) [0-9][0-9][0-9][0-9]-[0-9][0-9]-[0-9][0-9](T[0-9][0-9](:[0-9][0-9] (:[0-9][0-9](.[0-9]+)?(Z|[+-][0-9][0-9]:[0-9][0-9])?)?)?)?

UnitSpec: -nothing- | Identifier

Operators that exist in the Arden Syntax but not in GLIF Arden:

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Unary: sort, count [of], avg [of], median [of], sum[of], stddev [of], variance [of], no [of], slope [of], min [of], minimum [of], max [of], maximum [of], increase [of], decrease [of], percent increase [of], percent decrease [of], interval [of], arccos [of], arcsin [of], arctan [of], cos [of], cosine [of], sin [of], sine [of], tan [of], tangent [of], exp [of] (raises its argument to mathematical e), ceiling [of], truncate [of], log [of], log10 [of], abs [of], abs [of], sqrt [of], reverse [of], list [of], string [of], extract characters

Binary: merge, || , format, round, min...from, minimum...from, max...from, maximum...from, seqto

Operators that exist in GLIF_Arden but not in Arden Syntax:

Unary: from now, is unknown

Binary: in, not in (same as Arden's is in, is not in), overlaps, xor, |* , is a, is-a, occur/occurs/occurred at, at least...of

Attributes

didactics

encoding_language

specification

Attribute Detail

didactics

Data type: Supplemental_Material
Multiplicity : 0..*
Description : supplemental information about the expression
Level: A, B, C

encoding_language

specification

Data type: string
Multiplicity: 0:1
Description:a structured string written in GLIF_Arden that specifies the expression
Level: B, C

Class *Expression_Package.List_Expression*

Inherits from:

Expression_Package.Expression

List_Expression: an expression involving comma-separated strings, numbers, variable names, or other literals

Class *Expression_Package.Numerical_Expression*

Inherits from:

Expression_Package.Expression

Class *Expression_Package.Probabilistic_Criterion*

Inherits from:

Expression_Package.Criterion

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

To add conditional probabilities or Bayesian network info

Class *Expression_Package.String_Expression*

Inherits from:

Expression_Package.Expression

String_Expression: an expression involving strings

Class *Expression_Package.Three_Valued_Criterion*

Inherits from:

Expression_Package.Criterion

Interface Detail

Interface *Expression_Package.Expression_Language*

Attributes

GLIF_Arden

Other

SQL3

Attribute Detail

GLIF_Arden

Other

SQL3

Interface *Expression_Package.Three_Valued_Type*

Attributes

false_value

true_value

unknown_value

Attribute Detail

false_value

true_value

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

unknown_value

Example:

(smoker and chronic cough) Three_Valued_Criterion

name: three-valued criterion that assesses whether a patient is a smoker and has a chronic cough encoding_language: GLIF_Arden specification: (smoker=true) AND (chronic_cough=true) didactics

(smoker and chronic cough) Probabilistic_Criterion

name: probabilistic criterion that assesses whether a patient is a smoker and has a chronic cough encoding_language: GLIF_Arden specification: (smoker=true) AND (chronic_cough=true) didactics

The three-valued_criterion specification: (smoker = true) AND (chronic_cough = true) evaluates to true if a patient is a smoker and has a chronic cough. It evaluates to false if the patient is not a smoker, or does not have a chronic cough, or both. It evaluates to unknown if it is unclear that the patient is a smoker or has a chronic cough, or both.
As a specific example, if we don't know whether or not the patient is a smoker and we know that the patient has a chronic cough, the criterion evaluates to unknown.
To evaluate the probabilistic_criterion specification: (smoker = true) AND (chronic_cough = true), we need to know the probability that a patient is a smoker, P(smoker=true), the probability that a patient has a chronic cough, P(chronic_cough=true), and the probability that a patient has a chronic cough given that the patient is a smoker, P(chronic_cough=true| smoker=true). This last probability expresses the dependence relationship between a patient being a smoker and the patient having a chronic cough. Given these three probabilities, the specification evaluates to a real number in the interval [0,1]: P((smoker = true) AND (chronic_cough = true)) = P(smoker=true)* P(chronic_cough=true| smoker=true)
If there is a 70% probability that the patient is a smoker and we know that there is a 95% probability of a patient having a chronic cough given that s/he is a smoker, then, P((smoker = true) AND (chronic_cough = true)) = .7 * .95 = 0.665

Package *Guideline_Package*

Classes

class Guideline_Package.Algorithm

class Guideline_Package.Guideline

class Guideline_Package.Maintenance_Info

class Guideline_Package.Nestable

Interfaces

interface Guideline_Package.Guideline_Status

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Class Diagram

Interface Node Detail

Interface *Guideline_Package.Guideline_Status*

Class Detail

Class *Guideline_Package.Algorithm*

Inherits from:

Guideline_Model_Entity

Attributes

first_step

steps

Attribute Detail

first_step

steps

Class *Guideline_Package.Guideline*

Inherits from:

Guideline_Package.Nestable

A guideline has slots for defining the algorithm, maintenance information, eligibility criteria, intention of the guideline, didactics,

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

exceptions for leaving the guidelines, and the default viewer.

Attributes

algorithm

data_items

default_viewer

didactics

eligibility_criteria

exceptions

intention

maintenance_info

Attribute Detail

algorithm

data_items

default_viewer

Data type: string
Multiplicity: 0:1
Description:the default viewer of the guideline
Level: A

didactics

eligibility_criteria

exceptions

Data type: Exception
Multiplicity: 0:*
description: the set of exceptions that upon their occurrence a new next step is entered.
Level: B

intention

maintenance_info

Class *Guideline_Package.Maintenance_Info*

Attributes

author

guideline_status

modification_date

version

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Attribute Detail

author

guideline_status

modification_date

version

Class *Guideline_Package.Nestable*

Inherits from:

Guideline_Model_Entity

A Nestable is an object that can be used to create an iterative or nested specification of a guideline step. Currently, the Guideline and Macro classes are the sub-classes of Nestable.

Interface Detail

Interface *Guideline_Package.Guideline_Status*

Attributes

obselete

published

unpublished

Attribute Detail

obselete

published

unpublished

Package *Guideline_Step_Package*

Subpackages

package Guideline_Step_Package.Decision_Step_Package

package Guideline_Step_Package.Macro_Package

Classes

class Guideline_Step_Package.Action_Step

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

class Guideline_Step_Package.Branch_Step

class Guideline_Step_Package.Guideline_Step

class Guideline_Step_Package.Patient_State_Step

class Guideline_Step_Package.Synchronization_Step

Interfaces

interface Guideline_Step_Package.Order_Constraint

interface Guideline_Step_Package.Selection_Method

Class Diagram

Package Node Detail

Package *Guideline_Step_Package.Decision_Step_Package*

Package *Guideline_Step_Package.Macro_Package*

Interface Node Detail

Interface *Guideline_Step_Package.Order_Constraint*

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Interface *Guideline_Step_Package.Selection_Method*

Class Detail

Class *Guideline_Step_Package.Action_Step*

Inherits from:

Guideline_Step_Package.Guideline_Step

Action Steps specify clinical actions that are to be performed in the patient-care process. An action step specifies a set of tasks (Action_Specifications) that need to be performed. An action step has only one next-step. The action step has attributes that specify its iteration information, duration, triggering events, and associated exceptions. Action Steps are nested by including a Subguideline_Action type of task in the step. The Subguideline_Action task has a (sub)guideline attribute that contains the nested subguideline.

When a guideline step finished its execution and the control flow is about to pass to the next step, then, if the next step has associated triggering events, then this next step is executed only after one of its triggering event occurred.

Attributes

duration_constraint

exceptions

iteration_info

next_step

tasks

triggering_events

Attribute Detail

duration_constraint

Data type: Time_Literal_Expression
Multiplicity: 0:1
description: An interval shose start time is the shortest time that the step should take and whose end time is the maximum time that a step could take.
Level: B

exceptions

Data type: Exception
Multiplicity: 0:*
description: the set of exceptions that upon their occurrence a new next step is entered.
Level: B

iteration_info

Data type: Iteration_Specifiation
Multiplicity: 0:1
Description: the iteration specification is compared
Level: A, B and C

next_step

Data type: Guideline_Step
Multiplicity: 1:*

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Description: The next step
Level: A, B

tasks

triggering_events

Data type: Event_Start_Time_Constraint
Multiplicity: 0:*
Description: set of events that can trigger the step and their associated timing constraints. If several triggering events are defined for a step, then the step can occur whenever one of the events occur, provided that you are in the previous step.
Level: B

Class *Guideline_Step_Package.Branch_Step*

Inherits from:

Guideline_Step_Package.Guideline_Step

Description:
Branch steps deirect flow to multiple *guideline steps*. All of these *guideline steps* must occur in parallel. A *branch step* may link a *guideline step* to any other *guideline step*.

Purpose:
The *branch step* is used to model concurrent *guideline steps*. The selection method and order constraint attributes that previously characterized the *branch step* were removed so that the *branch step* would not semantically overlap the *case* and *synchronization* steps.

Considerations:
The selection method attribute that previously characterized the *branch step* were removed so that the *branch step* would not semantically overlap the *case* and *synchronization* steps.

Attributes

branches

order_constraint

Attribute Detail

branches

Data type: Guideline_Step
Multiplicity: 1:*
Description: an unordered list of subsequent guideline steps that should occur in parallel.
Level: A, B

order_constraint

Data type: Order_Constraint
Multiplicity: 1:*
Description: specifies whether the branches should be done in parallel or in any order
Level: A, B

Class *Guideline_Step_Package.Guideline_Step*

Inherits from:

Guideline_Model_Entity

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Attributes

didactics

Attribute Detail

didactics

Class *Guideline_Step_Package.Patient_State_Step*

Inherits from:

Guideline_Step_Package.Guideline_Step

Description:
A Patient State step (scenario) is a guideline step (a node in the flowchart) that is used for two purposes. One is as a label that describes a patient state that is achieved by previous steps. This way, a guideline may be viewed as a state transition graph, where states are scenarios, or patient states, and transitions between these states are the networks of guideline steps (excluding patinet state steps) that occur between two patinet state steps. The other purpose of a patient state step is an entry point to the guideline (e.g., patient came back to the clinic at state A).

A patient state step has a criterion that describes the state of the patient who is at that patient state. A patient state step is followed by a guideline step.

When a patient arrives at a clinic, his current state is compared to the last patient state that was recorded for him. If he is not at that state, then the patient state steps that represent new encounters are searched. These can be determined either by an implementation-level attribute called "new_encounter" of type boolean, which characterizes a patient state step (implementation level attribute) or by looking at patient state steps whose next-step is triggered by an event of type "new patient encounter".

Purpose:
as defined above.

Considerations:
1) When a patient arrives at a clinic he is matched to one of the p[ossible Patient State Steps. If none fit, we try to track his expected current state going forward from the last scenario that the patient was at. It is important to acknowledge the fact that a patient might not follow the guideline precisely, and that he/she may be treated also outside the regular clinic.

2) In Samson's EON guideline model, a patient state step has a reference advice guideline that offers advice as to what tests should be done to further evaluate the patient's state. This advice guideline is different from a regular guideline in 3 respects. (a) it is not required that this advice guideline be followed; (b) it does not include activities; (3) the steps of the advice guideline. This special reference advice, or consultation guideline, is used in order to avoid using the branch step. Having a consultation guideline as an attribute of a patient state step overlaps in functionality with the branch step. It allows the possibility that the physician sees the recommendations listed under the consultation guideline but can follow these recommendations (or not) up until the time that the patient enters another scenario (see example ERD). We sould like to use the branch step, and so, we do not need to use consultation guidelines, and will stick to using just one kind of guideline for now.
3) We commented out the new_encounter attribute. The guideline author would probably not want to specify whether a patient state step represents a new encounter or not; this would probably be institution specific, and needs to be determined during localization/ implementations.
4) Instead of specifying that a patient state step represents a new encounter, we can alternatively specify a triggering event of type: "patient encounter" that will trigger the next step pointed to by the patient state step.
3) If there is a criterion that refers to a generalization (e.g., "state is not well") it also applies to specializations of that class (e.g., "state is fever"). The hierarchy of terms is defined in the domain ontology.

Attributes

label

next_step

patient_state_description

Attribute Detail

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

label

Data type: String
Multiplicity: 1
Description: defines the patient state in text that refers to terms taken from the domain ontology
Level: B
Example:
While the patient_state_description can be "Temperature > 99 F", the label can say "fever".

next_step

Data type: Guideline_Step
Multiplicity: 0:1
Description: The step that follows the scenario, if its criterion evaluates to true
Level: B

patient_state_description

Data type: Criterion
Multiplicity: 1
Description: the expression that defines the patient state
Level: B

Class *Guideline_Step_Package.Synchronization_Step*

Inherits from:

Guideline_Step_Package.Guideline_Step

Synchronization steps are used in conjunction with branch steps. When a branch step is followed by multiple guideline steps, the flow of control must eventually converge in a single step. Each branch may lead to a series of steps, resulting in a set of branching paths. The step at which the paths converge is the synchronization step. When the flow of control reaches the 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.

Attributes

attribute1

continuation

next_step

Attribute Detail

attribute1

continuation

Data type: Selection Method
Multiplicity: 1
Description: specifies whether the next step should wait for all, some or one of the branches to finish execution.
Level: A, B

next_step

Data type: Guideline_Step
Multiplicity: 1:*
Description: The next step
Level: A, B

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Interface Detail

Interface *Guideline_Step_Package.Order_Constraint*

Attributes

anyOrder

parallel

Attribute Detail

anyOrder

parallel

Interface *Guideline_Step_Package.Selection_Method*

Attributes

all_of

one_of

some_of

Attribute Detail

all_of

one_of

some_of

Example:

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

In this example, the branch step, named "Do All" is shown on the top. There are 3 branches, which are all action steps: randomization, baseline study, and Stratification. The selection method specifies that all of these action steps should be performed following the branch step, and the order constraint specifies that they should be done in parallel.

Example:

(Samson: With Consultation Guideline)

Example:

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

(MOR) with branch step

Package *Guideline_Step_Package.Decision_Step_Package*

Classes

class Guideline_Step_Package.Decision_Step_Package.Case_Condition

class Guideline_Step_Package.Decision_Step_Package.Case_Step

class Guideline_Step_Package.Decision_Step_Package.Choice

class Guideline_Step_Package.Decision_Step_Package.Choice_Step

class Guideline_Step_Package.Decision_Step_Package.Decision_Condition

class Guideline_Step_Package.Decision_Step_Package.Decision_Option

class Guideline_Step_Package.Decision_Step_Package.Decision_Step

class Guideline_Step_Package.Decision_Step_Package.KofN_Choice

class Guideline_Step_Package.Decision_Step_Package.Operator

class Guideline_Step_Package.Decision_Step_Package.RuleIn_Choice

class Guideline_Step_Package.Decision_Step_Package.Utility_Choice

class Guideline_Step_Package.Decision_Step_Package.Utility_Choice_Step

class Guideline_Step_Package.Decision_Step_Package.Weighted_Criterion

Interfaces

interface Guideline_Step_Package.Decision_Step_Package.Operator_Type

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Class Diagram

Note:
options[i].condition_value of type Case_Condition

Note:
options[i].condition_value of type Choice

Note:
choice is of class UtilityChoice

Interface Node Detail

Interface *Guideline_Step_Package.Decision_Step_Package.Operator_Type*

The operator "in" is used for intervals. For example, "x in (3,5), means x> 3 and x <5.

Class Detail

Class *Guideline_Step_Package.Decision_Step_Package.Case_Condition*

Inherits from:

Guideline_Step_Package.Decision_Step_Package.Decision_Condition

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Attributes

case_value

Attribute Detail

case_value

Data type: Expression
Multiplicity: 1
Description: the value of an expression to which the Case_Step.case_expression is compared
Level: B

Class *Guideline_Step_Package.Decision_Step_Package.Case_Step*

Inherits from:

Guideline_Step_Package.Decision_Step_Package.Decision_Step

Description:
Case steps represent a deterministic decision between guideline steps. The different decision options are mutually exclusive. When a decision option's condiiton value matches the case step's condition, then controll flows to the destination specified by that decision option.

IMPORTANT:
options.condition_value must be of class CaseDestination.

Purpose:

Considerations:
1.The name of the case step will probably change when we compete the decision hierarchy.

Note:
options[i].condition_value of type Case_Condition

Class *Guideline_Step_Package.Decision_Step_Package.Choice*

Inherits from:

Guideline_Step_Package.Decision_Step_Package.Decision_Condition

Description:
Choices are used by *choice steps* to direct flow from one *guideline step* to another. A *choice* may be one of several kinds: *RuleInRuleOutChoice*, *KofNChoice*, and *UtilityChoice*. There may be other kinds of choices depending on how these choices are ranked.

Class *Guideline_Step_Package.Decision_Step_Package.Choice_Step*

Inherits from:

Guideline_Step_Package.Decision_Step_Package.Decision_Step

Description:
Choice steps represent a non-deterministic decision between guideline steps. The different decision options are not necessarily mutually exclusive. Ranking the decision options depends on the class of choices. In a ruleInRuleOutChoice, for example, when a decision option's condition value's rule in and strict rule in matches the choice step's condition, then control may flow to the destination specified by that decision option. (1) If the criterion matches more than one of a set of values specified by the choices, a

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

person will have to determine the next guideline step at run-time. Only one guideline step will be chosen to be the next step to be executed. (2) Each option contains a degree of preference that may be modeled differently for the different types of choices. (3) The degree of preference will determine how the choices will be ranked. This will assist the user in choosing among the different options.

IMPORTANT:
options.condition_value must be of class Choice.

Purpose:

Considerations:

Note:
options[i].condition_value of type Choice

Class *Guideline_Step_Package.Decision_Step_Package.Decision_Condition*

Description:
Decision conditions are classified as either case destinations or choices depending on whether the decision step that contains it is a case step or a choice step, respectively. It contains a name and a value that determines the next step for a case step or the rank of a choice for a choice step.

Attributes

name

Attribute Detail

name

Data type: String
Multiplicity: 1
Description: the name of a case destination or choice
Level: A,B,C

Class *Guideline_Step_Package.Decision_Step_Package.Decision_Option*

Description:
Decision Options are used by *Decision steps* to direct flow from one *guideline step* to another. Each *decision option* consists of a *Decision Condition* and a destination *guideline step*. The *condition* of a *decision step* is compared to the *condition value* of a *decision option*. If they are equal, then the control flows to the *guideline step* that is specified by the *decision option*.

Attributes

condition_value

destination

Attribute Detail

condition_value

Data type: Decision_Condition
Multiplicity: 1
Description: the value of an expression to which the Decision_Step.expression is compared
Level: B

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

destination

Data type: Guideline_Step
Multiplicity: 1
Description: the destination step that is taken when a Decision_Step.expression equals the Decision_Option.destination
Level: A, B

Class *Guideline_Step_Package.Decision_Step_Package.Decision_Step*

Inherits from:

Guideline_Step_Package.Guideline_Step

Description:
Decision steps direct flow from one *guideline step* to another. A *decision step* may link a *guideline step* to any other *guideline step*. A *decision step* contains a condition, which is an expression. The *condition's* value determines the control flow to one of a set of possible *guideline steps*, which are specified by the *options* of the *decision step*. The *condition* is compared, using an *operator* to the *options.condition_value*. If the *condition* matches one of the *decision options* then the control can flow, in the case of a choice step, and must flow, in the case of case step, to the *guideline step* that is specified by that *decision option's destination*. If the *condition* does not match any of the set of values specified by the *decision options*, or, if available data do not allow evaluation of the *condition*, then the control flows to the default destination *guideline step*.

When a guideline step finished its execution and the control flow is about to pass to the next step, then, if the next step has associated triggering events, then this next step is executed only after one of its triggering event occurred.

Purpose:
The *conditional step* used an extended Boolean model. This made it cumbersome and error-prone to represent criteria that do not have a true-or-false outcome (e.g., what is the patient's age category: neonate, infant, toddler, child, adolescence, adult, elderly). Therefore, the *case step* replaced the *conditional step* by allowing a conditional choice to be made among several alternative *guideline steps*.

Considerations:
1.The operator was added so that different comparisons would be possible. For example the operator may be "=", or may be "in" to represent a comparison to an interval (age in (30,40) as opposed to age eq > 30)

Attributes

condition

decision_detail

default_next_step

duration_constraint

exceptions

iteration_info

operator

options

triggering_events

Attribute Detail

condition

Data type: Expression
Multiplicity: 1
Description: the expression whose value is compared to the different decision options
Level: B

decision_detail

Data type: Nestable

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Multiplicity: 0:1
Description: This optional attribute is used for specifying a nested condition. The execution of this nested item (a subguideline or macro) would precede the evaluation of the decision to be made by the step. The nested object would create or modify *"variables"* that are used in the decision criterion.
Level: B

default_next_step

Data type: Guideline_Step
Multiplicity: 0:1
Description: the step that follows when none of the choices or case destinations are feasible
Level: B

duration_constraint

Data type: Time_Literal_Expression
Multiplicity: 0:1
description: An interval shose start time is the shortest time that the step should take and whose end time is the maximum time that a step could take.
Level: B

exceptions

Data type: Exception
Multiplicity: 0:*
description: the set of exceptions that upon their occurrence a new next step is entered.
Level: B

iteration_info

Data type: Iteration_Specifiation
Multiplicity: 0:1
Description: the iteration specification is compared
Level: A, B and C

operator

Data type: Operator
Multiplicity: 1
Description: a comparison operator used to compare the case expression to the different case values of the case step's destinations
Level: B

options

Data type: Decison_Option
Multiplicity: 1:*
Description: the list of conditions and their corresponding destination steps
Level: B

triggering_events

Data type: Event_Start_Time_Constraint
Multiplicity: 0:*
Description: set of events that can trigger the step and their associated timing constraints. If several triggering events are defined for a step, then the step can occur whenever one of the events occur, provided that you are in the previous step.
Level: B

Class *Guideline_Step_Package.Decision_Step_Package.KofN_Choice*

Inherits from:

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Guideline_Step_Package.Decision_Step_Package.Choice

Description:
KofNChoices contain an array of criteria, each associated with a weight. The weighted criteria for each of the options will determine how an option will be ranked among the choices presented to a user at run-time. The sum of the weights for each criterion in a choice has to equal 1.0. The higher the value of a choice (from 0 to 1), the higher its rank.

Attributes

criterion_and_weight

Attribute Detail

criterion_and_weight

Data type: Weighted_Criterion
Multiplicity: 1:*
Description: the list of criteria and weights that establish a numerical value for this choice
Level: B

Class *Guideline_Step_Package.Decision_Step_Package.Operator*

The operator is used to compare the case expression in a decision step to the actual case value in a case step.

Attributes

operator

Attribute Detail

operator

Data type: Operator_Type
Multiplicity: 1
Description: used to compare a case expression with a case value
Level: A,B,C

Class *Guideline_Step_Package.Decision_Step_Package.RuleIn_Choice*

Inherits from:

Guideline_Step_Package.Decision_Step_Package.Choice

Considerations:
1) It is possible to add two more attributes: rule_out and strict_rule_out, but we can always use the "not" operator (e.g., rule_in: not A or rule_out: A), and so the rule_out and strict rule_out are not required.

Example: Strict-rule-in criterion : Penicillin allergy=false vs. Strict-rule-out criterion: Penicillin allergy= true for giving amoxicillin prophylaxis for bacterial endocarditis.

2) If the criterion language will distinguish between criteria and strict criteria, then we can use just the rule_in attribute. The question is whether this destinction is required elsewhere, or just in making choices.

Attributes

rule_in

strict_rule_in

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Attribute Detail

rule_in

Data type: Criterion
Multiplicity: 1
Description: the value may either be true or false and plays a role in ranking this choice
Level: B

strict_rule_in

Data type: Criterion
Multiplicity: 0:1
Description: the value may either be true or false and plays a role in ranking this choice as best choice if true or worst choice if false
Level: B

Class *Guideline_Step_Package.Decision_Step_Package.Utility_Choice*

Inherits from:

Guideline_Step_Package.Decision_Step_Package.Choice

Description:
UtilityChoices contain the node that will determine it's expected value or expected utility at run time. This expected utility will determine its rank among the list of choices.

Attributes

Node

Attribute Detail

Node

Data type: String
Multiplicity: 1
Description: the expected value of a choice based on a decision analysis model
Level: B

Class *Guideline_Step_Package.Decision_Step_Package.Utility_Choice_Step*

Inherits from:

Guideline_Step_Package.Decision_Step_Package.Choice_Step

Description:
Utility Choice steps represent a choice step that uses the Utility theory in deciding among several options. It contains a pointer to the actual algorithm used to evaluate the choices. This may either be a decision analysis tree or an influence diagram.

IMPORTANT:
options.condition_value must be of class UtilityChoice.

Note:
choice is of class UtilityChoice

Attributes

DecisionDiagram

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Attribute Detail

DecisionDiagram

Data type: URL
Multiplicity: 1
Description: the location of the utility diagram/decision analysis model
Level: A,B,C

Class *Guideline_Step_Package.Decision_Step_Package.Weighted_Criterion*

The weighted criterion contains the case value or the expression to be compared against the actual case expression in a decision step. This is assigned a weight that together with the actual numerical case value determines the rank of a particular choice.

Attributes

criterion

weight

Attribute Detail

criterion

Data type: CriterionL
Multiplicity: 1
Description: the criterion may evaluate to true or false
Level: B

weight

Data type: int
Multiplicity: 1
Description: the weight given to a criterion
Level: A,B,C

Interface Detail

Interface *Guideline_Step_Package.Decision_Step_Package.Operator_Type*

The operator "in" is used for intervals. For example, "x in (3,5), means x> 3 and x <5.

Attributes

equals

in

Attribute Detail

equals

in

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Example:

In this example, the case step, named "Breast Mass Type" is shown on the top. There are 4 possible destinations, depending on the value of the criterion BMT. If BMP is equal to "no mass" then the destination is the case step "patient age > 30?"; If BMP is equal to "solid mass" then the destination is the action step "mammogram"; If BMP is equal to "cystic mass" then the destination is the action step "aspirate cyst "; If BMP is equal to "indeterminate mass" then the destination is the action step "Ultrasound"

Example:

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Example:

This is based on the American Heart Association's prevention of Bacterial Endocarditis Guideline. (Prophylactic Regimens for Dental, Oral, Respiratory Tract, or Esophageal Procedures).

The case step in example 2 is a soft case step and may actually be modeled as a choice step. The criteria in a case step are all absolute. One may be able to take oral meds, but still opt to take Ampicillin IV.

Package *Guideline_Step_Package.Macro_Package*

Classes

class Guideline_Step_Package.Macro_Package.Arden_Macro

class Guideline_Step_Package.Macro_Package.Macro

class Guideline_Step_Package.Macro_Package.Macro_Action

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

class Guideline_Step_Package.Macro_Package.Macro_Decision_Step

class Guideline_Step_Package.Macro_Package.Macro_Schema

class Guideline_Step_Package.Macro_Package.Macro_Action_Step

Class Diagram

Note:
This step can contain only one task and its type must be Macro_Action or its subclasses.

Note:
The decision_detail attribute of the Macro Decision Step must be of type Macro.

Class Detail

Class *Guideline_Step_Package.Macro_Package.Arden_Macro*

Inherits from:

Guideline_Step_Package.Macro_Package.Macro

Attributes

action

condition

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

evoke

5.1 Attribute Detail

action

condition

evoke

Class *Guideline_Step_Package.Macro_Package.Macro*

Inherits from:

Guideline_Package.Nestable

Attributes

schema

Attribute Detail

schema

Data type: Macro_Schema
Multiplicity: 1
Description: The schema description of the GLIF pattern and mapping of attributes of the macro to their values in the content_step.
Level: B

Class *Guideline_Step_Package.Macro_Package.Macro_Action*

Inherits from:

Action_Specification_Package.Action_Specification

Attributes

macro

Attribute Detail

macro

Class *Guideline_Step_Package.Macro_Package.Macro_Action_Step*

Inherits from:

Guideline_Step_Package.Action_Step

Description:
Many guidelines or their sections can often be represented using a *pattern* of GLIF steps and components. For example, a clinical trial protocol consists of phases, each phase consists of visits, during a visit, the trial coordinator usually performs certain actions (orders tests, delivers treament) and completes case report forms. A risk assessment guideline consists of data collection steps, followed by risk calculation, followed by recommendations that are based on the risk profile. In fact, even an Arden Syntax MLM , could be described using a pattern of GLIF components: an event (trigger), followed by a decision step (logic slot), followed by

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

action steps (action slot). The pattern of GLIF varies in the description of certain parameters in each of these examples. In the case of the Arden Syntax MLM, these are the actual event, decision, and actions in that instance of the MLM.
A macro step is a (potentially though not necessarily nested) guideline step that describes such patterns of GLIF components in a custom step. The attributes of the macro step are the parameters of the pattern of GLIF components. The attributes of a *Clinical Trial Visit* macro step are the time of the visit, the tasks to be performed during the visit, the forms to be completed, etc.
Using macro step to represent commonly occuring patterns has benefits for authoring, visual understanding, and execution of guidelines. For example, using macro steps in a protocol authoring tool can make it easier to edit protocols. The author has to deal withconcepts that are at the domain-level (such as visits, forms, etc.) rather than creating flowcharts with action steps, branch steps, and decision steps.
Note that *Macro Step* is an abstract class. Attributes of a particular type or class of Macros would be defined in a subclass of *Macro Step*.
The GLIF pattern of a Macro Step is described in the Macro Schema object that is referenced by the step. This pattern maps attributes of the macro to components in a GLIF step. Thus, using the schema and the values of attributes, a complete sequence of GLIF objects can be constructed. Because the knowledge of the macro can be transformed to a sequence of GLIF components, even applications that are unaware of particular macros and their attributes, can display and execute the underlying GLIF. In this way, macros can be used to create application-specific components with minimal effect on sharability of GLIF.

Purpose:
To develop custom/application-specific extensions of GLIF steps.

Considerations:
The design of an object model for macros has two challenges

- Creating a general macro step that could be distinguished as an action or a decision, while using single inheritance in the class diagram. The current design can distinguish the macro as being an action or a decision by examining the class of the step described in the schema (schema.glif_pattern). Thus, if this step is an action step, the macro is an action. If this step is a decision step, the macro is a decision step.

- We would like to create a formal method to describe the pattern of GLIF components (including the permitted variations) and the mapping of attributes of the macro to their value descriptions in the content step. One possible way to do so, is to use a language similar to the Schema representation of XML. Much work is required to develop such a language. We defer the development of the macro schema language until later. The Macro_Schema class provides a placeholder.

Class *Guideline_Step_Package.Macro_Package.Macro_Decision_Step*

Inherits from:

Guideline_Step_Package.Decision_Step_Package.Decision_Step

Note:
The decision_detail attribute of the Macro Decision Step must be of type Macro.

Class *Guideline_Step_Package.Macro_Package.Macro_Schema*

Description:
A macro schema is an abstract description of a particular class of macros. The schema describes the GLIF structure/pattern of a Macro and the mapping between the GLIF structure and the attributes of the macro.
A GLIF pattern describes the sequence of GLIF steps and other GLIF components that define the contents of the macro.

Purpose:
The schema defines the mapping of a macro to the underlying GLIF. The GLIF pattern is used to generate instances of GLIF objects that can be executed or viewed as such. The schema is also used for validation of macro attributes.
The schema consists of a single action step or decision step. The action step or decision step may contain a nested subguideline of any complexity.

Considerations:
A notation/language for schemas still has to be developed. This language will be used to describe schemas abstractly. The definition of this language will require significant effort, which we will defer until other aspects of GLIF are more fully developed.
We are considering a notation similar to the XML schema definition syntax for the World Wide Web Consortium (W3C).
The examples provided in this document are only an indication of our current ideas on how GLIF schemas will be described. These are not intended to be specifications.

Attributes

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

glif_pattern

macro_class_name

Attribute Detail

glif_pattern

Data type: Object (to be defined)
Multiplicity: 1
Description: The description of the GLIF pattern in a notation to be defined.
Level: B

macro_class_name

Data type: String
Multiplicity: 1
Description: The class of macros that are encoded by this instance of Macro_Schema, e.g., "Clinical Management State" from GEODE.
Level: A

Example:

```
<action_step id="*">
<action_detail>
<Guideline>
<algorithm>
<steps>
<decision_step>
<event>
<macro_attribute name="evoke">
<macro_attribute_constraints>
</macro_attribute_constraints>
</macro_attribute>
</event>
<condition>
<macro_attribute name="condition">
</macro_attribute>
</condition>
<options>
<case_destination>
<case_value>true</case_value>
<destination><action_step ref="x"/></destination>
</case_destination>
</options>
</decision_step>
<action_step id="x">
<macro_attribute name="action">
</macro_attribute>
</action_step>
</steps>
</algorithm>
</Guideline>
</action_detail>
<next_step>
</next_step>
</action_step>
```

This is an example of a Macro ("MLM Macro") that illustrates how an Arden Syntax MLM can be described with a GLIF Macro. The Arden Macro object (a subclass of MacroStep) has 4 attributes, 3 of which map to MLM attributes, and one attribute which links in the next step. The three MLM attributes are

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

1. Evoke
2. Condition
3. Action

Note: This may also need a data slot.

Note:
true

Example:

This MLM macro from MLM Macro 1 is expanded into a full GLIF object.

Note:
Event attribute of macro

Note:
Condition attribute of macro

Note:
Action attribute of macro

Example:

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

A generic risk assessment guideline contains three parts or "steps". Each of these steps can be modeled as a macro.

1. **Collecting patient data** - Data that is needed for calculating risk is collected in this step. The data may be obtained conditionally based on values of previously collected data. In this example, demographics data is obtained for all patients. Menstrual history is obtained only if the condition adult female is true.
   The data collection step is modeled using an ordered list of Get_Conditional_Data macros. This macro contains a condition and a list of patient data items that must be obtained.

2. **Computing risk** - The risk calculations are performed in this step. This macro has to be defined. It would contain definitions of variables that are to be created and the calculation of those variables in Assignment_Actions.

3. **Recommendations** - Recommendations based on computed risk and individual risk factors are provided in this step. Recommendations are provided only if an associated condition is true. The Exercise_recommendation is provided only to high risk persons.
   Thus, the Recommendations_Macro is structurally similiar to the Get_data_macro.

Note:
To be defined

Example:

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

This example expands the Risk Assessment Macro into its full GLIF implementation. That is, the macro has been expanded into more atomic GLIF steps.

Note:
To be defined

Package *Iteration_Package*

Classes

class Iteration_Package.Iteration_Specification

Class Diagram

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

The Iteration_Specification class specifies information regarding the loop structure of the iteration. The action- and decision steps that should be iterated, that references the Iteration_Specification, are iterated until the abort condition or stopping condition criteria hold. The iterations are carried out at a certain (fuzzy) frequency (e.g., every 8h +/- 30min or, 3 times a day). If the iteration does occur within the specified frequency, then the iteration time points remain according to schedule, and are not reset (e.g., the patient came in after 8 1/2 hours. The next iteration should take place after 7 1/2 hours). Some frequencies may have a specified allowed offset. This means that if the iteration began outside the fuzzy frequency, but within the offset, then the iteration takes place and the iteration points are reset. For example, the patient came back after 9 hours. He gets treated (since the offset is 8 +/-2 h) but the next iteration is 8 hours late.

Class Detail

Class *Iteration_Package.Iteration_Specification*

Inherits from:

Guideline_Model_Entity

Description:
The *Iteration_Specification* class specifies information regarding the loop structure of the iteration. The thing to be iterated, that references the *Iteration_Specification*, is iterated until the abort condition or stopping condition *criteria* hold. The iterations are carried out at a certain(fuzzy) frequency (e.g., every 8h +/- 30min or, 3 times a day). If the iteration does occur within the specified frequency, then the iteration time points remain according to schedule, and are not reset (e.g., the patinet came in after 8 1/2 hours. The next iteration should take place after 7 1/2 hours). Some frequncies may have a specified allowed offset. This means that if the iteration began outside the fuzzy frequency, but within the offset, then the iteration takes place and the iteration points are reset. For example, The patient came back after 9 hours. He gets treated (since the offset is 8 +/-2 h) but the next iteration is 8 hours late.

Purpose:
To specify iteration information **Considerations:**
1. Executing an action at irregular intervals can be either modeled as an iteration or not. For example, a visit schedule that says: "visit every 5 weeks for 5 months and then every 2 weeks" is an iteration, but an immunization schedule may be modeled as an action (Immunization) that is event triggered (and not as give shot every 1 month for month, thaen every 2 months for 2 months, etc.).
2. Sometimes one wishes to give 1-2 pills every 4-5 hours, but no more than 8 pills within 24 hours. The iteration is still every 4-5 hours, but the dose that is determined for each iteration is dependant on previous doses and may be equal to zero.

Attributes

abort_condition

frequency

stopping_condition

Attribute Detail

abort_condition

Data type: Criterion
Multiplicity: 0:1
Description: the expression under which the iteration is aborted. A value of -1 means iterate forever
Level: B

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

frequency

Data type: Temporal_Expression
Multiplicity: 0:1
Description: the frequency is either vague, e.g., 3 times a day, or fully specied, i.e., every 8 hours
Level: B and C

stopping_condition

Data type: Expression
Multiplicity: 0:1
Description: the expression under which the iteration should terminate. A value of -1 means iterate forever.
Level: B

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Examples of Iteration Specifications including loose, tight and discrete temporal criteria

(a) Iterate 10 days every 8 hours.
(b) Iterate 10 days 3 times a day.
(c) Iterate 30 times 3 times a day.
(d) Iterate 30 times every 8 hours.
(e) Iterate according to a discrete specification.

Package *Nesting*

Class Diagram

Nesting

Nesting of guideline steps is supported in the folowing manner:

Action Steps are nested by including a Subguideline_Action type of task in the step. The Subguideline_Action task has a subguideline attribute that contains the nested subguideline.

Decision Steps are nested by specifying a subguideline in the decision_detail attribute of the step. This subguideline is executed before the decision criterion for that step is evaluatedThe subguideline would modify or create new variables and assign them values. The use of these variables in the decision criteria makes the decision nested.

A nested action action or decision can potentially contain any kind of step including other actions, decisions, and other nested steps. Nesting is very useful for managing the complexity of guidelines. Nesting enables looking at a guideline from a top-level view and then zooming into/out of some of its parts. Nesting is also useful in representing a guideline in the context of other guidelines. Since nesting allows grouping of parts of a guideline into a single unit, this is a mechanism that can allow model extensibility and reuse of part of a guideline (defining macros), or adaptation of a guideline to a specific institution by replacing specifications for parts of a guideline (i.e., replacing a goal with a procedure). By use of nesting, steps will be defined as

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

containing other steps. As such, the guideline will be iteratively refined from a coarse level to a fine level of detail

Example:

This is an example of a nested Decision Step. The decision in Case_Step "Risk for IHD" is nested in the subguideline "IHD risk assessment".

This hypothetical subguideline determines risk for ischemic heart disease from physical exam, EKG, and stress testing. The leaf steps of this subguideline assign the risk value to a new data item named IHD_Risk using the Assignment_Action.

The data item risk_for_ihd that is created by the subguideline is used by the main case step in its case expression. The value is used by the case destinations to select the next step of the outer guideline.

This is an example of nesting an action step, for complexity management purposes.

Example:

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

This is an example of nesting an action step, for complexity management purposes.

Example:

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

This is an example of a nested Decision Step. The decision in Case_Step "Risk for IHD" is nested in the subguideline "IHD risk assessment".

This hypothetical subguideline determines risk for ischemic heart disease from physical exam, EKG, and stress testing. The leaf steps of this subguideline assign the risk value to a new data item named IHD_Risk using the Assignment_Action.

The data item risk_for_ihd that is created by the subguideline is used by the main case step in its case expression. The value is used by the case destinations to select the next step of the outer guideline.

This is an example of nesting an action step, for complexity management purposes.

Example:

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Package *Supplemental_Material_Package*

Classes

class Supplemental_Material_Package.Keyword

class Supplemental_Material_Package.Keyword_Didactic

class Supplemental_Material_Package.Local_Material

class Supplemental_Material_Package.Supplemental_Material

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

class Supplemental_Material_Package.WWW_Material

Class Diagram

Class Detail

Class *Supplemental_Material_Package.Keyword*

Attributes

concept_id

concept_name

vocabulary_id

Attribute Detail

concept_id

Data type: string
Multiplicity: 1
Description:the ID of the concept
Level: A<BR< Example: C0004057

concept_name

Data type: string
Multiplicity: 0:1
Description:the name of the concept
Level: A<BR< Example: Aspirin

vocabulary_id

Data type: string
Multiplicity: 1

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Description:the name of the vocabulary
Level: A<BR< Example: UMLS

Class *Supplemental_Material_Package.Keyword_Didactic*

Inherits from:

Supplemental_Material_Package.Supplemental_Material

Description:
A *Keyword_Didactic* is a list of keywords

Purpose: To allow a guideline_step or an entire guideline to reference keywords that can be used for search purposes

Considerations:

Attributes

keywords

Attribute Detail

keywords

Data type: Keyword
Multiplicity: 1:*
Description:the list of keywords
Level: A

Class *Supplemental_Material_Package.Local_Material*

Inherits from:

Supplemental_Material_Package.Supplemental_Material

Description:
Local_Material cis stored locally as text

Purpose:

Considerations:
Example:
bibliographic citations

Attributes

material

Attribute Detail

material

Data type: string
Multiplicity: 1
Description:the text representing the Local_Material
Level: A

Class *Supplemental_Material_Package.Supplemental_Material*

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Inherits from:

Guideline_Model_Entity

Description:
The *Supplemental_Material* class provides background or supporting information

Purpose:

Considerations:

Attributes

MIME_Type

Attribute Detail

MIME_Type

Data type: string
Multiplicity: 0:1
Description:Multipurpose Internet Mail Extensions (MIME) extends the format of Internet mail to allow non-US-ASCII textual messages, non-textual messages, multipart message bodies, and non-US-ASCII information in message headers
Level: A Examples of MIME_Types are: text/plain, text/html, image/gif, image/jpeg, mov/qt, charset=us-ascii

Class *Supplemental_Material_Package.WWW_Material*

Inherits from:

Supplemental_Material_Package.Supplemental_Material

Description:
WWW_Material is a reference to a URL that designates a site on the World Wide Web

Purpose:

Considerations:
Example:
bibliographic citations

Attributes

url

Attribute Detail

url

Data type: string
Multiplicity: 1
Description:the URL of the WWW_Material
Level: A

Package *Temporal_Expression_Package*

Classes

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

class Temporal_Expression_Package.Atomic_Frequency

class Temporal_Expression_Package.Discrete_Temporal_Expression_Table

class Temporal_Expression_Package.Duration

class Temporal_Expression_Package.Duration_Expression

class Temporal_Expression_Package.Every_Expression

class Temporal_Expression_Package.Frequency_Expression

class Temporal_Expression_Package.Fuzzy_Duration

class Temporal_Expression_Package.Temporal_Expression

class Temporal_Expression_Package.Time_Interval

class Temporal_Expression_Package.Time_Interval_Expression

class Temporal_Expression_Package.Time_Literal

class Temporal_Expression_Package.Time_Literal_Expression

class Temporal_Expression_Package.Times_Expression

Interfaces

interface Temporal_Expression_Package.Time_Unit

interface Temporal_Expression_Package.Time_Zone

Class Diagram

Interface Node Detail

Interface *Temporal_Expression_Package.Time_Unit*

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

An enumerated type that enumerates different possible time units

Interface *Temporal_Expression_Package.Time_Zone*

Class Detail

Class *Temporal_Expression_Package.Atomic_Frequency*

Description:
An *atomic frequency* specifies the frequency at which something should occur and the duration after which all iterations should end

Purpose

Considerations:

Example:

Attributes

frequency

Attribute Detail

frequency

Data type: Fuzzy_Duration
Multiplicity: 1
Description:The frequency of the atomic frequency.
Level: B and C

Class *Temporal_Expression_Package.Discrete_Temporal_Expression_Table*

Inherits from:

Temporal_Expression_Package.Temporal_Expression

Description:
A *discrete temporal expression table* specifies a table who consist of rows of pairs of frequecies and durations (the pairs are Atomic frequencies). The order of rows is important. The rows are to be executed from the top of the table to its bottom.

Purpose
To model a list of discrete frequencies.

Considerations:

Example:
see patient every 5 weeks for 5 months, then, every 2 weeks for 1 month, and then every week for 1 month.

Attributes

frequency_list

Attribute Detail

frequency_list

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Data type:Atomic_Frequency
Multiplicity: 1:*
Description:The frequency table is a table of discrete frequencies and thier related duration
Level: B and C

Class *Temporal_Expression_Package.Duration*

Inherits from:

Guideline_Model_Entity

Description:
A *duration* has a value and a time unit

Purpose
To store a time value and its time unit

Considerations:

Example:
2 hour

Attributes

number

time_unit

Attribute Detail

number

Data type: float
Multiplicity: 1
Description:the value of the Duration
Level: A, B and C

time_unit

Data type: Time_Unit
Multiplicity: 0..1
Description:the time unit
Level: A, B and C

Class *Temporal_Expression_Package.Duration_Expression*

Inherits from:

Temporal_Expression_Package.Temporal_Expression

Class *Temporal_Expression_Package.Every_Expression*

Inherits from:

Temporal_Expression_Package.Frequency_Expression

Description:
An *every expression*specifies that something should occur every fuzzy duration with an allowed before and after offset

Purpose
In order to represent tight temporal expressions, as shown in the example

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Considerations:
We decided to comment the slot "repeat_every" of type Fuzzy_Duration, which is the structured way to specify an every expression, since we alreaqdy have the specification slot (of type string) that will store the duration expression.

Example:
Do something every 8 hours.

Class *Temporal_Expression_Package.Frequency_Expression*

Inherits from:

Temporal_Expression_Package.Temporal_Expression

Description:
A *tight or loose temporal expression* .

Purpose
In order to represent either a tight or a loose temporal expressions

Considerations:
We decided to comment the slot "for" of type Duration, which is the structured way to specify a duration expression, since we alreaqdy have the specification slot (of type string) that will store the duration expression.

Example:

Class *Temporal_Expression_Package.Fuzzy_Duration*

Inherits from:

Guideline_Model_Entity

Description:
A *fuzzy duration* is a duration that has an asociated before and after uncertainty. Any time point within (duration-berfore_uncertainty, duration+after_uncertainty) is considered to be within the duration of the fuzzy duration.

Purpose:
Sometimes a duration is not enough for our modeling purposes, and we need a fuzzy duration.

Considerations::
Example:
The patient should come back to the clinic after 2 weeks, with a before and after uncertainties of 2 days.

Attributes

after_offset

after_window

before_offset

before_window

duration

label

Attribute Detail

after_offset

Data type: Time_Magnitude
Multiplicity: 0:1
Description:The thing that is being iterated should iterate every "every" duration. If something occurred outside the "every" duration then it is NOT on time. If the occurrence happend in the interval (every.duration - every.before_uncertainty -

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

allowed_before_offset, every.duration - every.before_uncertainty) then the iteration point should be reset to the time of the event occurrence
Level: B and C

after_window

Data type: Time_Magnitude
Multiplicity: 0..1
Description:any time point within (duration, duration+after_uncertainty) is considered to be valid
Level: B and C

before_offset

Data type: Time_Magnitude
Multiplicity: 0:1
Description:The thing that is being iterated should iterate every "every" duration. If something occurred outside the "every" duration then it is NOT on time. If the occurrence happend in the interval (every.duration - every.before_uncertainty - allowed_before_offset, every.duration - every.before_uncertainty) then the iteration point should be reset to the time of the event occurrence
Level: B and C

before_window

Data type: Time_Magnitude
Multiplicity: 0..1
Description:any time point within (duration-before_uncertainty, duration) is considered to be valid
Level: B and C

duration

Data type: Time_Magnitude
Multiplicity: 1
Description:the basic unfuzzy duration
Level: B and C

label

Data type: string
Multiplicity: 0:1
Description:a text representation of the Iteration_Specification
Level: A

Class *Temporal_Expression_Package.Temporal_Expression*

Inherits from:

Expression_Package.Expression

Description:
temporal expressions concepts to be expressed within a guideline. A temporal expression consists of a label, a specification, and didactics inherited from the expression class

Purpose

Considerations:
Temporal expressions which are a function of the number of the iteration, or the time that passed from the first iteration, or the time that passed from some milestone, are represented only as discrete set temporal criterion, and not as a function such as 2i or 2(10-t).

In order for GLIF to include temporal expressions, there has to be a notion of time-stamped events or episodes and time-stamped clinical data/patient-related concepts and this currently does not exist. This could be captured by having variables that have associated start and end times to represent different events/episodes, etc. An instantaneous event would have equal start and end times and an episode with a duration would have an end time greater than the start time. Currently there is no means of analyzing temporal trends in GLIF, and it is probably something we want to support in future versions of GLIF. To facilitate analysis of temporal trends in the

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

future, we could introduce (now) the Arden method of creating a list that describes an event or episode's history with associated start and end time-stamps (e.g., a list of blood pressure readings for a patient at different instances in time where start_time=end_time for each reading).

Example:
every 8 hours
3 times a day

TemporalExpression:
TimeLiteral | Duration | TimeInterval | Frequency | FrequencySet | Duration UnaryOperator | UnaryOperator Number of Identifier | UnaryOperator Identifier | TemporalExpression BinaryOperator TemporalExpression | Identifier BinaryOperator TemporalExpression | TemporalExpression BinaryOperator Identifier | TimeInterval lasting Duration to Duration

Class *Temporal_Expression_Package.Times_Expression*

Inherits from:

Temporal_Expression_Package.Frequency_Expression

Description:
A *times expression* specifies that something should occur a specified number of times within a specified interval (the per attribute)

Purpose
In order to represent loose temporal expressions, as shown in the example

Considerations:
1) This class should not be used during execution. It should be mapped, or refined to a tight temporal expression.
2) We decided to comment the slots "per" of type Duration, and "repeat_times" of type integer, which is the structured way to specify a times expression, since we already have the specification slot (of type string) that will store the duration expression.

Example:
Do something 3 times a day.

Class *Temporal_Expression_Package.Time_Interval*

Attributes

end_time

start_time

Attribute Detail

end_time

Data type: Time_Literal
Multiplicity: 1
Description:
Level: A

start_time

Data type: Time_Literal
Multiplicity: 1
Description:
Level: A

Class *Temporal_Expression_Package.Time_Interval_Expression*

Inherits from:

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Temporal_Expression_Package.Temporal_Expression

); a time-interval_expression specifies an interval which is defined by a start_time and an end_time

Class *Temporal_Expression_Package.Time_Literal*

Attributes

day

hour

millisecond

minute

month

plus_hour

second

year

zone

Attribute Detail

day

hour

millisecond

minute

month

plus_hour

second

year

zone

Class *Temporal_Expression_Package.Time_Literal_Expression*

Inherits from:

Temporal_Expression_Package.Temporal_Expression

a time-literal_expression involves a specific instance in time (expressed as yyyy-mm-ddThh:mm:ss(Z|+/-hh) based on Arden syntax notation

Interface Detail

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Interface *Temporal_Expression_Package.Time_Unit*

An enumerated type that enumerates different possible time units

Attributes

DAY
HOUR
millisecond
MINUTE
MONTH
SECOND
YEAR

Attribute Detail

DAY

HOUR

millisecond

MINUTE

MONTH

SECOND

YEAR

Interface *Temporal_Expression_Package.Time_Zone*

Attributes

CST
ESTH
MT
PST
UTC

Attribute Detail

CST

ESTH

MT

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

- **PST**
- **UTC**

Example:

(a) An example of a fuzzy duration. A pill has to be taken every 24 hours. If the person forgot to take the pill, she may take it within 12 hours, otherwise, the dose is skipped.
(b) An example of offsets. A pill should be taken every 4-5 hours. The next pill should be taken 4-5 hours after the previous pill was taken.

Example:

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

The time-literal expression specification: 2 days BEFORE 1999-11-22T08:30:00 produces the result: 1999-11-20T08:30:00

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

which is of type time-literal.
The every_expression specification: every 4 hours with window -30 minutes, +30 minutes, offset -1 hour, +1 hour

The times_expression specification: 3 times a day for 2 years

Package *Views_Package*

Classes

class Views_Package.View_Specification

Class Diagram

Class Detail

Class *Views_Package.View_Specification*

Inherits from:

Guideline_Model_Entity

Description:

Note that any user should be able to see any entity in arbitrary detail, but only the specified user will see this level of detail by default
.

Purpose:

Views are a way of managing the complexity of guidelines.

If we accept the thesis that one purpose of guidelines is to reduce practice variability in favor of superior patterns, it seems likely that guidelines with greater scope will eventually appear.

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Views describe the amount of detail (i.e. level of nesting) displayed by default for each of the subguidelines.

If a guideline consists of Subguidelines, each of these subguidelines may be visualized as a triangle, with one step at the highest level and multiple steps at the lowest level. That is, the width of the triangle is proportional to the number of steps at that level of nesting. Top of the triangle = less detail = small number of steps. Bottom of the triangle = more detail = large number of steps.

A given filter (e.g. MD_Radiation_Oncologist) will define the default level of nesting/zooming for each of the subguidelines. It will be up to the guideline author to define the subguidelines in an appropriate way (e.g. to avoid too many steps per screen for a given viewer) and to define the level of nesting required for each given subguideline. In the above example, suppose that a breast cancer guideline has four subguidelines as above. A Radiation Oncologist looking at the guideline may see (by default) relatively little detail about screening, diagnosis and surgery. He will see a great deal of detail regarding radiation therapy, however. A surgeon looking at the same guideline may see little detail re: screening/diagnosis, a lot on surgery and little re: radiation therapy.

The status quo of specialty bodies publishing guidelines may change as multi-specialty organizations publish multi-specialty documents. Guidelines may become quite complex.

Much of medicine is multi-disciplinary in nature. The distinction between specialties is artificial. For example, the distinction between cardiology and nursing is for the convenience of practitioners. The patient suffering a myocardial infarction (heart attack) is likely to require care from both a cardiologist and a nurse. The information needs of the cardiologist, however, are very different from those of the nurse. The purpose of default views in GLIF should be to reveal to the cardiologist only the relevant portions of the myocardial infarction guideline, which may be different from that shown to the nurse.

Consideration:

Views are default filters through which we interact with the guideline. By definition, views do not change guideline logic (e.g. if an RN should do something different from an MD, this should be represented in the guideline logic, not in the view). Although we anticipate that the most common use of views will be user and/or location, there may be other relevant filters (e.g. situation such as routine vs. disaster). The view class is a guideline entity. Alternatively, the view could have been modeled as an enumerated type attribute. The main purpose of this class is to allow differential display in the simplest possible way.

The view specification was chosen to be at the level of guideline entities and not at the attribute level. We may later choose to make **attributes** (and not entire guideline entities) visible or invisible to some users.

Issues yet to be resolved:

1. Semantics of no default_user specified (e.g. confidential vs. visible to developer only).

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

2. Issues related to nesting?

Attributes

filter

Attribute Detail

filter

Data type: Filter_expression
Multiplicity: 0:1
Description: Views are a filter applied to the guideline, there may be multiple relevant filters, e.g. viewer (MD vs. RN), location (office vs. hospital), situation (disaster vs. routine), etc.

BNF:

term: filter_type = *domain_ontology_filter_instance*

filter_expression: term | *expression binary_operator expression* | *unary_operator expression* | **(***expression***)**

binary_operator: **OR** | **AND**
unary_operator: **NOT**

filter_type: **USER** | **LOCATION**

domain_ontology_filter_instance: **MD** | **RN** | ...

Level: A, B and C

Example:

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

A guideline might call for a breast biopsy. Lets say that all MDs want to see that a breast biopsy is called for, however, surgeons want to know what kind of biopsy is needed, incisional or excisional. .

Example:

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

This example shows how nesting deals with views. If the viewer is an MD he sees the top-level view of the action step Biopsy. He can zoom into the action-detail subguideline, to see that incisional or excisional biopsies can be performed.

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

An MD_Surgeon will directly see the zoomed-in view of biopsy directly, showing the decision that is made between incisional and excisional biopsy.

Note:
MD view:

Note:
MD_Surgeon view

Bibliography

American College of Cardiology/American Hospital Association/American College of Physicians-American Society of Internal Medicine (1999). "Guidelines for the Management of Patients with chronic Stable Angina." J Am Col Cardiol **33**: 2092-2197.

Boxwala, A. A., R. A. Greenes, et al. (1999). Architecture for a multipurpose guideline execution engine. Proc AMIA Symp 1999.

Clinical Decision Support & Arden Syntax Technical Committee of HL7 (1999). Arden Syntax for Medical Logic Systems, version 2.0. Draft revision. Health Level Seven. USA**:** 1-119.

Object Management Group (1999). The Common Object Request Broker: Architecture

and Specification.

Ohno-Machado, L., J. H. Gennari, et al. (1998). "The GuideLine Interchange Format: A Model for Representing Guidelines." Journal of the American Medical Informatics Association **5**(4): 357-372.

Ohno-Machado, L., S. J. Wang, et al. (1999). "Decision support for clinical trial eligibility determination in breast cancer." Proc Amia Symp.

Pattison-Gordon, E. (1996). ODIF:

Object Data Interchange Format. Boston, MA, Decision Systems Group, Brigham and Women's Hospital.

Shortliffe, E. H., G. O. Barnett, et al. (1996). "Collaborative medical informatics research using the Internet and the World Wide Web." Proc AMIA Annu Fall Symp: 125-9.

Zielstorff, R. D., J. M. Teich, et al. (1998). "P-CAPE: a high-level tool for entering and

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

processing clinical practice guidelines. Partners Computerized Algorithm and Editor." Proc Amia Symp: 478-82.

Appendix A Areas for future development

- Evaluation, testing, and refinement of GLIF3 specification
- Specification for goals and intentions of guideline steps
- Improved specification for temporal queries and expressions
- Better separation of high-level guidelines from implementation guidelines

Appendix B Examples

Screening for Thyroid Disease, ACP-ASIM Scientific Policy, Ann Intern Med. 1998; 129:141-143, code 500700180.
http://www.acponline.org/sci-policy/guidelines/

Recommendations

It is reasonable to screen women older than 50 years of age for unsuspected but symptomatic thyroid disease. The preferred screening method is a sensitive TSH test. A free thyroxine test should be done when the TSH level is undetectable or is 10 mU/L or more. Patients who have an undetectable TSH level and an elevated free thyroxine level have overt hyperthyroidism. Patients who have a TSH level higher than 10 mU/L and a low free thyroxine level have overt hypothyroidism. Patients with either of these conditions are likely to benefit from appropriate treatment.

The following two pages are screen shots of the GLIF specification of this guideline, done by Aziz Boxwala, using the GEODE Authoring Tool, Gauth.

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Guidelines for the Management of Patients with chronic Stable Angina, ACC/AHA/ACP-ASIM, Circulation 1999; 99:2829-2848, J Am Col Cardiol 1999; 33:2092-2197
http://www.acc.org/clinical/guidelines/june99/index.html

The following two pages are screen shots of the GLIF specification of this guideline, done by Mor Peleg and Elmer Bernstam, using the PROTÉGÉ Authoring Tool.

* For decisions marked with an asterisk, we assume that there is a list of contraindications available.

Figure 3. The Unstable Angina guideline: top-level

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Figure 4. Tasks done in the "Patient History and Physical" action step of Figure 3

Figure 5. Tasks done in the "Diagnostic tests" action step of Figure 3

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

StableAngina_INSTANCE_00383 (instance of ...

Specification

```
IntermediateRiskUnstableAngina OR
HighRiskUnstableAngina ==
(last rest_pain_duration > 20 minutes OR
 is in(Pulmonary_Edema, Problem_List) OR
 ( is in(Angina_At_Rest, Problem_List) AND last
ST_wave_change >= 1 mm ) OR
 ( is in(Angina, Problem_List) AND ( is
in(New_MR_Murmur, Problem_List) OR
        is in(Worsening_MR_Murmur, Problem_List) )) OR
  is in(Angina_With_S3, Problem_List) OR
  is in(New_rales, Problem_List) OR
  is in(Worsening_rales, Problem_List) OR
  is in(Angina_With_Hypotension, Problem_List))

OR

( (is in(rest_angina_duration > 20 minutes,
rest_angina_duration) AND
      last rest_angina_duration < 20 minutes AND
          (is in(moderate_likelyhood_of_CAD, Problem_List)
 OR
          is in(high_likelyhood_of_CAD, Problem_List)))  OR
  last rest_angina_duration > 20 minutes OR
  (is in(rest_angina, Problem_List) AND is in
(sublingual_nitroglycerin,
                    Rest_Angina_Relif_List)) OR
  is in(nocturnalAngina, Problem_List)  OR
  is in(NewOnsetCCSCIIIAngina, ProblemList) where ( now
- time of
       NewOnsetCCSCIIIAngina < 2 weeks) AND is
in(high_likelyhood_of_CAD,
            Problem_List))) OR
is in(NewOnsetCCSCIVAngina, ProblemList) where ( now -
time of
       NewOnsetCCSCIIIAngina < 2 weeks) AND is
in(high_likelyhood_of_CAD,
            Problem_List))) OR
is in(PathologicQWave, ProblemList) OR
last rest_ST_wave_change <= 1 mm  AND pattern is
(anterior OR inferior OR lateral)  OR
now-DateOfBirth > 65 years))
```

Figure 6. Specification of the criterion for the choice step "Intermediate or high risk unstable angina" of Figure 3

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Figure 7. Zoom-in view of the Treatment action step of Figure 3

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Figure 8. Zoom-in view of the "anti-anginal treatment" action step of Figure 7

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Figure 9. Zoom-in view of the "Beta-blocker therapy" action step of Figure 8

Figure 10. Specification of the criterion for the case step "Beta-blocker contraindications" of Figure 9

Figure 11. Zoom-in view of the "Add or substitute Ca++ Channel Blocker if no contraindication" action step of Figure 8

Figure 12. Specification of the criterion for the choice step "Ca++ Channel Blocker Contraindications" case step of Figure 11

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Figure 13. Zoom-in view of the "Add long-acting nitrate therapy" action step of Figure 8

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Figure 14. Zoom-in view of the "Education and risk factor modification of Figure 7

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Figure 15. Zoom-in view of the "Aspirin 81 to 325 mg QD" Action step of Figure 14

Figure 16. Zoom-in view of the "Clopidogrel or Wafarin" action step of Figure 14

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Figure 17. Zoom-in view of the "Routine Follow-up" Action step of Figure 8 and Figure 14

Figure 18. Zoom-in view of the "Diet" action step of Figure 18

Figure 19. Specification of the criterion for the "Need Diet?" choice step of Figure 18

Figure 20. Zoom-in view of the "Follow-up visits" action step of Figure 17

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Figure 21. Zoom-in view of the "Laboratory examination" of Figure 20

Figure 22. Iteration specification of the "Diabetes screening" action step of Figure 21

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Figure 23. Zoom-in view of the "ECG and Follow-up testing" action step of Figure 20

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Figure 24. Specification of the criterion for the "Do Stress Imaging?" choice step of Figure 23

Figure 25. Specification of the criterion for the "Do Treadmill testing?" choice step of Figure 23

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Figure 26. A zoom-in view of the "Consider revascularization therapy" action step of Figure 8

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.

Specification

```
PerformCABG == SifnificantLeftMAinCoronaryDisease OR
Three-Vessel_Disease OR (Two-Vessel_Disease AND
Significant_ProximalLeftAnteriorDescendingCAD AND
(AbnoramlLVFunction OR IschemiaOnNoninvasiveTesting)) OR
((One-Vessel_CAD OR Two-Vessel_CAD) AND
NoSignificantProximalLeftAnteriorDescendingCAD AND
(SurvivedSuddenCardiacDeath OR
SurvivedSustainedVentricularTachycarida)) OR
(not CABG AND (One-Vessel_CAD OR Two-Vessel_CAD) AND
no SignificantProximalLeftAnteriorDescendingCAD AND
LargeAreaOfViableMyocardium AND
HighRiskCriteriaOnNoninvasiveTesting) OR (no PTCA AND no
SuccessfulTreatmentByMEdicalTherapy AND
CanUndergoRevascularizationWithAcceptableRisk) OR
MultipleSaphenousVeinGraftStenoses OR (no PTCA AND
(One-Vessel_CAD OR Two-Vessel_CAD) AND no
SignificantProximalLADDisease AND ModerateAreaOfViable
Myocardium AND
DemonstartedIschemiaOnNoninvasiveTEsting) OR (no PTCA
ADN One-Vessel_Disease AND
SignificantProximalLADDisease)
```

Figure 27. specification of the criterion for the Perform CABG? Choice of Figure 26

```
PerformPTCA == ((Two-Vessel_Disease OR
Three-Vessel_Disease) AND
SignificantProximalLeftAnteriorDescendingCAD AND
AnatomicallySuitableForCathter-BasedTherapy AND
NormalLVFunction AND no TreatedDiabetes) OR
((One-Vessel_CAD OR Two-Vessel_CAD) AND
NoSignificantProximalLeftAnteriorDescendingCAD AND
(SurvivedSuddenCardiacDeath OR
SurvivedSustainedVentricularTachycarida))
 OR (no CABG AND no
SuccessfulTreatmentByMEdicalTherapy AND
CanUndergoRevascularizationWithAcceptableRisk) OR
MultipleSaphenousVeinGraftStenoses OR (no CABG AND
(One-Vessel_CAD OR Two-Vessel_CAD) AND no
SignificantProximalLADDisease AND
ModerateAreaOfViable Myocardium AND
DemonstartedIschemiaOnNoninvasiveTEsting) OR (no
CABG ADN One-Vessel_Disease AND
SignificantProximalLADDisease)
```

Figure 28. specification of the criterion for the Perform PTCA? Choice of Figure 26

Disclaimer: this is a draft version, not to be distributed or quoted. Copyright by Stanford University and Brigham and Women's Hospital.