Clinical Elements

Joey Coyle
Craig Parker
Stan Huff

May 24, 2005
# Contents

1 Clinical Data Models .......................................................... 7  
   1.1 Intro .............................................................................. 7  
   1.2 General Requirements ................................................... 10  

2 The Basics ........................................................................... 11  
   2.1 Introduction .................................................................... 11  
   2.2 Clinical Element Instances ............................................. 11  
   2.3 Key ............................................................................... 12  
   2.4 Data ............................................................................... 13  
   2.5 Items ............................................................................. 13  
   2.6 Qualifiers and Modifiers .................................................... 13  

3 CEML ................................................................................... 17  
   3.1 Introduction .................................................................... 17  
   3.2 Items ............................................................................... 18  
   3.3 Data ............................................................................... 19  
   3.4 Qualifiers ....................................................................... 20  
   3.5 Modifiers ...................................................................... 21  
   3.6 Modifier and Qualifier Scope ............................................ 21  

4 Modifiers and Qualifiers ......................................................... 23  
   4.1 Introduction .................................................................... 23  
   4.2 Modifiers ...................................................................... 23  
      4.2.1 Negation or Certainty Modifier ................................ 23  
      4.2.2 Subject Modifier ...................................................... 25  

5 Modifier and Qualifier Inheritance ........................................ 29  
   5.1 Introduction .................................................................... 29  
   5.2 Rules ............................................................................. 29
6 HL7 Version 3 Datatypes
6.1 Introduction .................................. 31
6.2 Description .................................... 32
  6.2.1 ANY ....................................... 32
  6.2.2 HL7:PQ ................................... 32
  6.2.3 HL7:ED ................................... 33
  6.2.4 HL7:ST ................................... 33
  6.2.5 HL7:CWE ................................. 33
  6.2.6 HL7:TS ................................... 34
  6.2.7 HL7:II ................................... 34
  6.2.8 HL7:REAL ................................. 35
  6.2.9 HL7:INT .................................. 35
  6.2.10 HL7:TEL ................................. 35
  6.2.11 HL7:BL .................................. 35
  6.2.12 HL7:CS .................................. 36
  6.2.13 HL7:CO .................................. 36
  6.2.14 HL7:IVL Types ............................ 37
  6.2.15 HL7:RTO Types ............................ 37
  6.2.16 Misc ASN.1 Declarations .................. 37
7 Glossary ........................................ 39
  7.1 Definitions ................................... 39
8 To Be Organized ................................. 41
  8.1 Versioning ................................... 41
  8.2 Errors ....................................... 41
  8.3 Linking screen fields to data fields .............. 41
  8.4 TypeOid ..................................... 42
  8.5 Tidbits ....................................... 42
  8.6 Interfaces ................................... 42
  8.7 ReportFlag ................................... 42
  8.8 Labels ........................................ 42
List of Figures

1.1 MLM snippet demonstrating Curly Braces. .................. 9
2.1 Relationship of CEML to Schema ............................. 11
2.2 Clinical Element Instance ..................................... 12
2.3 LabObservation with Key and Data Value ..................... 13
2.4 BloodPressure with Key and Item Values ..................... 14
2.5 Clinical Element with Mods and Quals ......................... 15
3.1 A simple Blood Pressure Panel with no Qualifiers .......... 18
3.2 Attributes of cedtype ......................................... 19
3.3 Attributes of the key Definition .............................. 19
3.4 Attributes of the item Definition ............................. 20
3.5 Values for cardinality ........................................ 20
3.6 A simple Systolic Blood Pressure definition with no Qualifiers 20
3.7 A simple Systolic Blood Pressure definition with Qualifiers 21
3.8 Attributes of the qual Definition .............................. 21
4.1 No Diagnosis of Breast Cancer ................................. 24
4.2 Mother had a Diagnosis of Breast Cancer ..................... 26
4.3 Mother had a Diagnosis of No Breast Cancer .................. 27
6.1 Physical Quantity. ............................................. 32
6.2 Encapsulated Data. ........................................... 33
6.3 Character String. ............................................. 33
6.4 Coded With Equivalents. ...................................... 34
6.5 Time Stamp. .................................................. 34
6.6 Instance Identifier. ........................................... 35
6.7 Real. ......................................................... 35
6.8 Integer Number. .............................................. 35
6.9 Telecommunication Address. ................................... 36
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.10</td>
<td>Boolean</td>
<td>36</td>
</tr>
<tr>
<td>6.11</td>
<td>Coded Simple</td>
<td>36</td>
</tr>
<tr>
<td>6.12</td>
<td>Code With Equivalents with operator</td>
<td>37</td>
</tr>
<tr>
<td>6.13</td>
<td>Interval Types used by IHC.</td>
<td>38</td>
</tr>
<tr>
<td>6.14</td>
<td>Ratio Types used by IHC.</td>
<td>38</td>
</tr>
<tr>
<td>6.15</td>
<td>Misc. ASN.1 Declarations</td>
<td>38</td>
</tr>
</tbody>
</table>
Chapter 1

Clinical Data Models

1.1 Intro

Detailed clinical models are the basis for retaining computable meaning when data is exchanged between heterogeneous computer systems. Detailed clinical models are also the basis for shared computable meaning when clinical data is referenced in decision support logic. Exactly what we mean by “detailed clinical models” and how they relate to the use of clinical data by computers will be described below.

There are a number of motives for exchanging clinical data between heterogeneous computer systems. Data can be exchanged between different computers within a facility or enterprise in order to make information available to clinicians at the point of care, with the goal of improving clinical decision making. Serology and culture results can be sent from clinical laboratories to public health departments as a means of detecting an epidemic or a bioterrorism attack. Health care providers that are participating in clinical trials of new medications or other therapies need to exchange clinical data as part of the research protocols. In all of these situations, the goal is not just to have the data available for humans to read and understand, but to have the data structured and coded in a way that will allow computers to understand and use the information.

The most common strategy for representing data that is sent between different computer systems is to send the data as name-value pairs (also known as entity-attribute-value triplets. For example, if the results of a hematocrit test were to be sent between two systems the data could be represented as:

- Test name = Hematocrit, Value = 44.1%

The Health Level 7 (HL7) and Digital Imaging and Communications in Medicine (DICOM) standards use this strategy, and the Logical Identifier Names and Codes
(LOINC) coding scheme was created in order to supply the “name” part of the
name-value pair. The use of standardized codes (and their associated computable
definitions) as the names of test results allows computers to use the information in
decision logic, outcomes research, and other clinical calculations.

Single valued measurements like a hematocrit result are represented easily in a
single name value pair as shown above. However, as soon as the clinical measure-
ment is slightly more complicated, variations in how the data can be represented
present themselves. For example, there are at least three ways of representing the
results of patellar deep tendon reflexes.

- A single name/code and value
  - Left patellar deep tendon reflex intensity is 2+
- Combination of two names/codes and values
  - Patellar deep tendon reflex intensity is 2+
    - Laterality is left
- Combination of three names/codes and values
  - Deep tendon reflex intensity is 2+
  - Body part is patella
  - Laterality is left

When the complex nature of the data allows these different options for repre-
sentation, it is important to understand that these representations are equivalent,
otherwise a computer processing the data will not recognize the alternative forms
and will fail to use the data appropriately. The ability of a computer to recog-
nize the equivalence of these statements is based on an underlying detailed clinical
model. For the example given, the detailed clinical model could be stated as:

- Type of measurement - (intensity of deep tendon reflex)
- Location of measurement - (patella, or patellar tendon)
- Laterality of measurement - (left side)

For a computer to recognize the equivalence of the three different statements,
there must be a more formal way of stating the information model and of referenc-
ing standardized terminologies that are used for the names of data elements in the
model. Use of standard models and associated standard reference terminologies
will enable a computer to detect equivalent representations.
Many examples of alternative data representation exist. A more difficult example than patellar reflex data is the problem of lung auscultation. The results of lung auscultation can be represented either in a finding focused style or a location focused style. For example, one can state the finding "wheezing", and then state every lung location where it was heard, such as "wheezing in the right and left upper lobes." Alternatively, one could be location focused, and state the location "right upper lobe", and state all the findings associated with that location, such as "the right upper lobe has wheezing, rales, and egophony."

The need for a formal way of representing detailed clinical models is closely related to what has been termed the “curly braces problem.” The curly braces problem arises from the practical issues of trying to implement medical logic modules (MLMs) such as rules, alerts, and reminders using Arden Syntax. Arden syntax is an HL7 standard for representing medical decision logic. In Arden syntax, data slots are used to create “read” statements that retrieve data that participates in the logic of the MLM from the patient’s EMR (or some other data store). Curly braces are used within the “read” statement “to isolate institution-specific portions [of data access] to one slot. Within the data slot, the institution-specific portions are placed in mapping clauses so that the institution-specific part does not interfere with the MLM syntax.” The following snippet from an MLM shows the use of curly braces.

```mlm
data:
/* creatinine in mg/dL*/
creatinine := read last {select value from lab where code = 237}
(more data declarations)
evoke
/* execute this logic each time a new calcium is stored*/
storage_of_calcium;
logic:
/* if creatinine is present and greater than 6, then stop now */
IF creatinine is present THEN
  IF creatinine is greater than 6.0 THEN
    conclude false
  ENDIF
ENDIF
...(more logic statements)
```

Figure 1.1: MLM snippet demonstrating Curly Braces.

In Figure 1.1, curly braces are used to enclose a SQL statement that would
retrieve the patient’s creatinine from a (hypothetical) relational database containing laboratory results. Pryor and Hripscak demonstrated that a major obstacle to sharing decision logic was creating the mapping from the logical data element in the read statement to the corresponding data in the local EMR. Detailed clinical models that incorporate standard coded terminologies are needed to overcome this problem. If these kinds of models existed and MLM’s referenced the common models, then a given institution would still need to map from the shared models to its local EMR representations, but once the mapping was completed, MLM’s from any source could be shared without further institution specific mapping.

The need for detailed clinical models has been recognized by researchers and standards organizations. DICOM, Centre for European Normalisation (CEN), Good Electronic Health Record (GEHR), HL7, GALEN, and Stephen Johnson have either developed or plan to develop a mechanism for describing and sharing detailed clinical models. In the following chapters we will discuss the development of IHC’s third generation clinical data model called the Clinical Element Model.

1.2 General Requirements

- The model must be comprehensive - it must accommodate representation of anything that can be stated about a patient.
- The model must be flexible and extensible - it must be possible to add elements and attributes to the model without requiring changes to underlying software and database.
- It must use an existing formalism (XML Schema, ASN.1, Conceptual Graphs, etc.) without modification.
- There must be a tight linkage to standard terminologies such as LOINC, Systematized nomenclature of medicine - Clinical Terms (SNOMED CT), HL7 Vocabulary Tables, etc.
- There must exist a mechanism to state negation, in order to say that something was NOT observed or was NOT present.
- A process for change management must be followed, in order to know which version of a model was in effect at the time data was stored.
- There is a need to easily change the cardinality of values; for example, to note a single complication versus selecting all the complications that apply.
- There must exist the ability to allow any degree of arbitrary collections and batteries.
- It must be possible to retain as part of the permanent record how the data was originally seen by the user, or as it was sent by an application.
Chapter 2

The Basics

2.1 Introduction

The Clinical Element is a recursive data structure designed to be the core building block in the design of detailed clinical data models. In this section, we will examine the attributes of a Clinical Element instance, and then later we will learn about the Clinical Element Definition Language (CEML) which is used to author the constraint definitions for a given instance. The CEML definition language is analogous to XML Schema for an XML instance document. See Figure 2.1.

![Figure 2.1: Relationship of CEML to Schema](image)

2.2 Clinical Element Instances

In the most basic skeleton of a Clinical Element, Figure 2.2, every Clinical Element has a key and a value. The key is a coded value represented by an HL7 version 3 datatype CWE. The value is a choice between a data element or items, where data is an HL7 version 3 datatype ANY, and items is a list of one or more Clinical Elements which gives the model its recursive nature.
2.3 Key

The Key is a coded value represented by an HL7:CWE. The code represents an important real world concept that is important to the Clinical element. This vagueness is what makes the key confusing to many.

The key serves to link different Clinical Elements types via a common real world concept, without the need for multiple concepts. As an example of this, IHC stores lab results in a type called LabObservation, but does not create special keys for each lab analyte. In Figure 2.3 is an example of LabObservation, which is a laboratory result for a serum sodium. The Key can use our code for SerumSodium rather than creating a special Key such as SerumSodiumLabResult. By using the code SerumSodium in LabObservation and in other types such as LabOrder, we have a common link between these types.¹

Where most people get confused by Key is in the Clinical Elements where it serves less of a purpose, such as when instead of a domain of possible Keys for a given type, there is only one possible value for Key. If it is a one to one map between a type and it’s key, then why do we require it in the instance? The answer is to remain consistent, and since there is always the possibility that, for any given type, we increase the possible number of values allowed in the Key from only one to a domain of values. If this occurs, we are still OK as long as the instances were stored with the correct Key.

¹One can argue that the LabObservation example in Figure 2.3 can be modeled differently, with a single key meaning LabObservation, and the analyte could be stored elsewhere in a Qualifier. We will talk about this later.
Rules for Key

- Key is required in Instances.
- Key always links to real world concept space.
- Possible Keys can only be restricted in subtypes, and thus not extended.
- Key should not be used as a query key alone, it must always be used in conjunction with the type.

2.4 Data

If a Clinical Element is designed to use data instead of items, this Clinical Element becomes a leaf node in the tree. It is in data where an HL7 version 3 datatype will be stored that will contain values such as numbers, strings, and codes. See Figure 2.3 as an example.

2.5 Items

If a Clinical Element is designed to use items rather that data, this Clinical Element becomes an internal node in the tree. The list of items in this Clinical Element, are each themselves Clinical Elements which can be either leaf nodes or internal nodes. An Example of of a ClinicalElement that uses items is seen in Figure 2.4.

2.6 Qualifiers and Modifiers

In addition to Key and the Value Choice, are two lists of Clinical Elements which serve to alter the meaning of the Value Choice. These two lists of Clinical elements are called Qualifiers and Modifiers, and they are named to represent the extent to which they alter the meaning of the Value Choice. See Figure 2.5. A Clinical
Element in Modifier alters the Value Choice to such an extent that one can never use the Value without considering the effect of the Modifier. Qualifiers, on the other hand, are considered to add information to a Value and don’t actually change the meaning of the Value in a way that makes it dangerous to ignore this change.²

²Whether or not qualifiers can actually ever be truly ignored is debated in informatics circles.
Figure 2.5: Clinical Element with Mods and Quals
Chapter 3

CEML

3.1 Introduction

In order to model the many Clinical Element types, a modeling language is required. One of the mistakes made by IHC in the past was to use a standard modeling language, but alter it in such a way that no standard tools could work with this language. This was IHC’s experience with ASN.1, which didn’t quite work for our needs, so additional constructs were added to the language. This immediately meant only IHC’s compiler and other tools would work with the source code definitions of our types. This defeated the purpose of using a standard modeling language in the first place. The question then arises as to why use a standard modeling language which carries the baggage of generalism, when you gain none of the benefits.

For the Clinical Element Model it was decided we would use a standard modeling language but not alter the language, to guarantee we could work with standard tools. XML Schema was chosen to be the formalism for Clinical Elements, and instances would exist as XML.\(^1\) The first thing that happened was a realization that the verbosity and complexity of XML Schema made the models difficult to comprehend and write, especially for the seemingly simple constructs we wanted to represent. It was decided we would create a schema shorthand language that we could use to generate the final XML Schemas. This new language was called Schemita, using the Spanish diminutive form of Schema.

Later we began to realize, why are we limiting our instances to one serializa-

\(^1\)It was also a hope we could move to an XML database, but the immaturity of this technology and people’s uncomfortableness with performance, combined with an immediated need to produce, put this on the back burner.
tion such as XML. The instances could exist as ASN.1 BER or even serialized Java Objects. And the Schemita could be used to generate the definitions in these different languages.

Eventually the name was changed to be called CEML for Clinical Element Markup Language, breaking the direct tie to XML Schema, and it has evolved from there.

### 3.2 Items

To understand how to define a Clinical Element with CEML it is probably easiest to just give a few simple examples and describe the example. Each definition in CEML is an XML document which describes the required parts for the Clinical Element type we are trying to describe. First lets start by describing a type for a Blood Pressure Panel in Figure 3.1. Remember that in Clinical Elements the value choice can contain either a list of child Clinical Elements or an HL7 datatype. In this first example, we create a type where the value choice contains two children; one for systolic blood pressure and one for diastolic blood pressure.

```xml
<cedtype name="BloodPressurePanel" base="ObservationPanel">
  <key code="BloodPressurePanelKey_NCID"/>
  <item name="systolicBloodPressure" type="SystolicBP"/>
  <item name="diastolicBloodPressure" type="DiastolicBP"/>
</cedtype>
```

Figure 3.1: A simple Blood Pressure Panel with no Qualifiers

Let’s examine the parts of this Clinical Element Definition in Figure 3.1. Every definition is defined using a `cedtype` element as the root element; `cedtype` is an abbreviation for Clinical Element Definition Type. The `name` attribute of `cedtype` identifies the name of this Clinical Element. The `base` attribute identifies the parent type of this Clinical Element. You can think of this Clinical Element as a subtype or subclass of the `base` type which in this case is ObservationPanel. So this new type we have defined called BloodPressurePanel will inherit the properties of the

---

2This was driven by the fear of only being able to use XML instances which are ten times bigger in size than a corresponding ASN.1 BER instance. How would this affect bandwidth and storage requirements?

3And even if we were only using XML as our sole serialization, then what about different forms of XML. We have already discovered that it is useful to have more than one XML Serialization for different usages such as storage versus for the developer. Which Schema would be the master?
CHAPTER 3. CEML

3.3. DATA

type ObservationPanel. A list of the allowable attributes allowed in \texttt{cedtype} can be seen in Figure 3.2.

- \textbf{name} - The name of this type being declared.
- \textbf{base} - The parent type from which this type is derived.
- \textbf{author} - A comma delimited list of usernames who have authored this cedtype.
- \textbf{status} - An authorship status, such as proposed.

Figure 3.2: Attributes of cedtype

Next, inside the \texttt{cedtype}, other parameters can be declared using various elements. In our example in Figure 3.1, there are two types of elements within the definition. The first is a \texttt{key} definition and the second is two \texttt{item} definitions.

In the \texttt{key} definition, the allowable key code for an instance can be restricted to a particular code, as it is here using the \texttt{code} attribute. An alternative to using the \texttt{code} attribute, is to use the \texttt{domain} attribute which restricts the allowable key code in an instance to be within a certain domain of codes. The attributes of the \texttt{key} definition are listed in Figure 3.3.

- \texttt{code} - Restricts all instances of this type to have this key code.
- \texttt{domain} - Restricts all instances of this type to have a key code within this domain.

Figure 3.3: Attributes of the key Definition

Next in our BloodPressurePanel example in Figure 3.1, there are two \texttt{item} definitions. These \texttt{item} definitions declare the child Clinical Elements that are required or allowed in the BloodPressurePanel we are defining. Each of these \texttt{item} definitions has a \texttt{name} attribute and a \texttt{type} attribute. The \texttt{type} attribute defines the Clinical Element type of this child, and the \texttt{name} attribute assigns a local name to this type. This is just like the normal variable-type assignments used in most programming languages. The attributes available in an \texttt{item} definition are listed in Figure 3.4. An additional attribute, not shown in our BloodPressurePanel example is the \texttt{cardinality} attribute, which defines the number of this child type that can occur in an instance. The available values for cardinality are shown in Figure 3.5.

3.3 Data

In the last example we saw a Clinical Element defined where the value choice was two child Clinical Elements rather than an HL7 datatype. So in this next example
3.4 Qualifiers

The examples in Figures 3.1 and 3.6 only declared definitions of the key and value choice (data or items). Next in Figure 3.7, we have added qualifier declarations pertinent to the systolic blood pressure. These qual definitions have the same attributes as the item definitions, with the addition of the scope attribute. See Figure 3.8. Scope will be explained in a later section.

These qualifiers add information about the measured systolic blood pressure, such as indicating the body position of the patient during the measurement, and what method or device was used to make the the systolic blood pressure measurement.
CHAPTER 3. CEML

3.5. MODIFIERS

```xml
<cedtype name="SystolicBP" base="Observation">
  <key code="SystolicBPKey_NCID"/>
  <data type="pq"/>
  <qual name="measurementMethodOrDevice" type="BPMeasurementMethodOrDevice"/>
  <qual name="bodyPosition" type="BodyPosition"/>
  <qual name="intravascularBodySiteExpr" type="IntravascularBodySiteExpr"/>
  <qual name="breathingPhase" type="BreathingPhase"/>
</cedtype>
```

Figure 3.7: A simple Systolic Blood Pressure definition with Qualifiers

<table>
<thead>
<tr>
<th>name</th>
<th>The local name of this qual.</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>The Clinical Element type of this qual.</td>
</tr>
<tr>
<td>cardinality</td>
<td>Restricts the number of occurrences for this qual.</td>
</tr>
<tr>
<td>scope</td>
<td>Controls this qual's inheritance to child Clinical Elements.</td>
</tr>
</tbody>
</table>

Figure 3.8: Attributes of the qual Definition

3.5 Modifiers

To Do ...

3.6 Modifier and Qualifier Scope
Chapter 4

Modifiers and Qualifiers

4.1 Introduction

As previously mentioned, Modifiers and Qualifiers both alter the meaning of the data in the Clinical Element. Modifiers do this to such an extent, that they actually significantly change the meaning of the data, and hence we say they “modify” the data. Qualifiers change the meaning of the data, but to a much lesser extent, and thus we say they “qualify” the data.

4.2 Modifiers

Modifiers are themselves Clinical Elements nested within a Clinical Element. Modifiers are considered dangerous, because they significantly change the meaning of the Clinical Element. The data in a Clinical Element with a modifier can not even be considered without also considering the impact any Modifiers have on that data. The most easily understood modifiers are those that cause negation.

4.2.1 Negation or Certainty Modifier

Originally the model contained a modifier for negation that was called Negation. In further review of this issue, it was determined there are really two shades of negation. These two shades of Negation are called CertaintyOfExistance and CertaintyOfOccurance. Both of these are children of a supertype called Certainty which is an HL7:CWE.

CertaintyOfExistance. The degree of certainty that what is in data is true. The range of values are “No, Probably Not, Maybe, Mightbe, Probably, Affirmative”
4.2. MODIFIERS

CHAPTER 4. MODIFIERS AND QUALIFIERS

CertaintyOfOccurrence. The degree of certainty that a procedure or action was performed. The range of values are “Absolutely Not, Unlikely, May have, Mightbe, Absolutely has”.

The absence of certainty defaults to Absolutely Certain. The Negation modifiers are only allowed in leaf node Clinical Elements. Leaf Node Clinical Elements are those that have an HL7 datatype in contrast to Clinical Elements that contain a list of Clinical Elements. In Figure 4.1 is an example of a statement of a Diagnosis of No Breast Cancer.

```
<DiagnosisAndFinding code="123" kind="item">
  <key>
    <code>123</code>
    <displayName>DiagnosisAndFinding</displayName>
  </key>
  <data>
    <cwe>
      <code>123</code>
      <displayName>BreastCancer</displayName>
    </cwe>
  </data>
  <CertaintyOfExistance code="123" kind="mod">
    <key>
      <code>123</code>
      <displayName>CertaintyOfExistance</displayName>
    </key>
    <data>
      <cwe>
        <code>123</code>
        <displayName>No</displayName>
      </cwe>
    </data>
  </CertaintyOfExistance>
</DiagnosisAndFinding>
```

Figure 4.1: No Diagnosis of Breast Cancer.

1These values need to be made consistent between the two
2I am concerned about the meaning of No Diagnosis of Breast Cancer Was Made, and how that relates to Diagnosis of No Breast Cancer
4.2.2 Subject Modifier

Another important modifier is called Subject. This modifier indicates who the data in the Clinical Element is about. In the absence of a Subject Modifier, Subject defaults to self. This modifier allows us to add Clinical Elements in a patient’s record that contain data about family members, household members, or donors. In Figure 4.2 is an example of a Diagnosis of Breast Cancer for the patient’s mother.

Subject can describe the Relationship, as well as a direct oid to a specific person with a unit number, an externally defined donor id.

Modifiers can also be combined to create more complex statements. In Figure 4.3 is an example of combining Certainty with Subject to state the Mother of the patient had a diagnosis of No Breast Cancer.
Figure 4.2: Mother had a Diagnosis of Breast Cancer.
Figure 4.3: Mother had a Diagnosis of No Breast Cancer
Chapter 5

Modifier and Qualifier Inheritance

5.1 Introduction

It has been stated that Qualifiers and Modifiers alter the meaning of the contents in the Value Choice. This is understandable when the Value Choice is Data and thus an HL7 datatype, but what does it mean when the Value Choice is Items and thus a list of Clinical Elements. The simple answer is that these qualifiers and modifiers are inherited by the children in items, and the qualifiers and modifiers are applied to their Value Choice.

5.2 Rules

The rules for modifier and qualifier inheritance from a parent to it’s children inside of items, depend on the value of the scope attribute in the CEML definitions of the modifiers and qualifiers.

Values for scope

- normal
- additive
- local

The default value is normal which is to be assumed in the following rules unless otherwise stated.
Chapter 6

HL7 Version 3 Datatypes

6.1 Introduction

In order to understand the Clinical Element Model, one must have some understanding of the HL7 version 3 datatypes. These datatypes are very important to the IHC’s model: one could actually think of the Clinical Element Model as an organizing container of HL7 version 3 datatypes. As part of HL7’s RIM, a set of datatypes called the HL7 version 3 datatypes have been defined. These datatypes are used to represent actual information such as numbers, strings, and codes. A UML and textual description of the HL7 datatypes is available from HL7. HL7 has implemented the datatypes in XML schema, and is now working with Sun Microsystem to develop them in Java.\(^1\)

In the following sections is a description IHC’s implementation of the version 3 datatypes in Abstract Syntax Notation 1 (ASN.1). This is a subset of the datatypes, but some of the datatype have additional attributes we found necessary for a permanent datastore.\(^2\) In addition, it should be understood that the HL7’s UML description is a hierarchy of types. IHC’s ASN.1 implementation is not a hierarchy but a flat declaration of the types with the appropriate attributes. To understand this section one should have some familiarity with the HL7 datatypes before reading.

\(^1\)HL7’s UML description describes methods and properties, which are both implemented in the Java types, yet the XML Schema only implements the properties.

\(^2\)It must be remembered, that the HL7 messages were not designed to be a permanent datastore; they were designed to be a temporary exchange format between disparate systems.
6.2 Description

6.2.1 ANY

ANY is the base datatype in the HL7 hierarchy and all other datatypes derive from this type. In IHC’s ASN.1 implementation, we actually do not declare this type, but instead just directly include its only attribute, called nullFlavor, in the other types.

6.2.2 HL7:PQ

Physical Quantity (Figure 6.1) is used to store a real number value with a real physical unit such as 2.3 milligrams, 24 days, or 2 drops. One thing to note is that HL7 requires that units be a UCUM code, but at IHC we have decided to use units from our own coding system. The UCUM system enables one to create physical quantities that are not constrained to any particular unit, but by using UCUM codes you could never create physical quantities such as 2 drops.

```
PQ ::= SEQUENCE {
  value [0] REAL-Value ,
  operator [1] CS-Operator OPTIONAL ,
  units [2] CS-Units OPTIONAL ,
  translations [3] SEQUENCE OF PQT OPTIONAL }
PQT ::= SEQUENCE {
  value [0] REAL-Value OPTIONAL ,
  operator [1] CS-Operator OPTIONAL ,
  code [2] Code OPTIONAL ,
  codeSystem [3] CS-CodeSystem OPTIONAL ,
  displayName [5] VisibleString OPTIONAL ,
  originalText [6] VisibleString OPTIONAL ,
  codingRationale [7] CS-CodingRationale OPTIONAL ,
  context [8] CS-Context OPTIONAL ,
  transient [9] CS-Transient OPTIONAL }
```

Figure 6.1: Physical Quantity.
6.2.3 HL7:ED

Encapsulated Data (Figure 6.2) can convey any data from a plain character string, formatted text, to multimedia data.

```
ED ::= SEQUENCE {
  mediaType [0] CS--MediaType OPTIONAL,
  language [1] CS--Language OPTIONAL,
  compression [2] CS--Compression OPTIONAL,
  integrityCheck [3] OCTET STRING OPTIONAL,
  integrityCheckAlgorithm [4] CS--IntegrityCheckAlgorithm OPTIONAL,
  reference [5] TEL OPTIONAL,
  thumbnail [7] ED OPTIONAL,
  value [6] OCTET STRING OPTIONAL }
```

Figure 6.2: Encapsulated Data.

6.2.4 HL7:ST

Character String (Figure 6.3) is used to store text/plain data. In the HL7 hierarchy it is a subtype of ED restricted to inline text/plain data.

```
ST ::= SEQUENCE {
  value [0] VisibleString }
```

Figure 6.3: Character String.

6.2.5 HL7:CWE

Coded With Exceptions (Figure 6.4) is used to store coded values. For a storage model, we have decided translations will occur very often, and we will always use CWE for coded values. Due to the performance requirements of a permanent storage based system, only codes from IHC’s coding system will be allowed in “code”, and alternate codes from other coding systems will be placed in translations. Each code will have an assigned “displayName” which will be stored in the instance. This binding of the codes with a default displayName is intended to improve retrieval performance, since it will not always be necessary to also retrieve surface forms for every code.
### 6.2. DESCRIPTION  
#### CHAPTER 6. HL7 VERSION 3 DATATYPES

<table>
<thead>
<tr>
<th>CWE ::= SEQUENCE {</th>
</tr>
</thead>
<tbody>
<tr>
<td>code [0] Code OPTIONAL,</td>
</tr>
<tr>
<td>displayName [1] VisibleString OPTIONAL,</td>
</tr>
<tr>
<td>originalText [2] VisibleString OPTIONAL,</td>
</tr>
<tr>
<td>codingRationale [3] CS-CodingRationale OPTIONAL,</td>
</tr>
<tr>
<td>translations [4] SEQUENCE OF CET OPTIONAL }</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CET ::= SEQUENCE {</th>
</tr>
</thead>
<tbody>
<tr>
<td>code [0] Code OPTIONAL,</td>
</tr>
<tr>
<td>codeSystem [1] CS-CodeSystem OPTIONAL,</td>
</tr>
<tr>
<td>codeSystemVersion [2] VisibleString OPTIONAL,</td>
</tr>
<tr>
<td>displayName [3] VisibleString OPTIONAL,</td>
</tr>
<tr>
<td>originalText [4] VisibleString OPTIONAL,</td>
</tr>
<tr>
<td>codingRationale [5] CS-CodingRationale OPTIONAL,</td>
</tr>
<tr>
<td>context [6] CS-Context OPTIONAL,</td>
</tr>
<tr>
<td>transient [7] CS-Transient OPTIONAL }</td>
</tr>
</tbody>
</table>

Figure 6.4: Coded With Equivalents.

### 6.2.6 HL7:TS

Point in Time (Figure 6.5) is used to store a point in time. Operator was added by IHC to represent concepts such as “greater than” or “less than”.

<table>
<thead>
<tr>
<th>TS ::= SEQUENCE {</th>
</tr>
</thead>
<tbody>
<tr>
<td>value [0] GeneralizedTime,</td>
</tr>
<tr>
<td>operator [1] CS-Operator OPTIONAL }</td>
</tr>
</tbody>
</table>

Figure 6.5: Time Stamp.

### 6.2.7 HL7:II

Instance Identifier (Figure 6.6) is used to uniquely identify an instance, thing or object.  

---

We need to discuss root, codes, and oids
6.2.8 HL7:REAL

Real Number (Figure 6.7) is used to store a real number values such as 1.25678, 3, or 0. IHC has added operator to represent concepts such as “greater than” or “less than”.

```
XREAL ::= SEQUENCE {
  value [0] REAL-Value,
  operator [1] CS-Operator OPTIONAL }
```

Figure 6.7: Real.

6.2.9 HL7:INT

Integer Number (Figure 6.8) is used to represent an integer number ranging from negative to positive infinity. Operator was added by IHC to represent codes such as “greater than” or “less than”.

```
XINT ::= SEQUENCE {
  value [0] INTEGER,
  operator [1] CS-Operator OPTIONAL }
```

Figure 6.8: Integer Number.

6.2.10 HL7:TEL

Telecommunication Address (Figure 6.9) is used to represent a resource mediated by telecommunication equipment.

6.2.11 HL7:BL

Boolean (Figure 6.10) is used to store a value of “true” or “false”.

```
6.2. DESCRIPTION

6.2.12 HL7:CS

Coded Simple (Figure 6.11) is simply a code and a textual name for display. All the CS-Foo types are just restrictions of CS where the code is restricted to a certain domain of values.\(^4\) We do not use CS in the data section of Clinical Elements, and instead only use them to define parts of the internal model.

\[
\text{CS ::= SEQUENCE } \{
\text{ code } [0] \text{ Code },
\text{ displayName } [1] \text{ VisibleString OPTIONAL }
\}
\]

\[
\text{CS−CodeSystem ::= CS } \text{−−TODO: domain}
\]

\[
\text{CS−Context ::= CS } \text{−−TODO: domain}
\]

\[
\text{CS−Transient ::= CS } \text{−−TODO: domain}
\]

\[
\text{CS−Operator ::= CS } \text{−−TODO: domain}
\]

\[
\text{CS−NullFlavor ::= CS } \text{−−TODO: domain}
\]

... etc ...

Figure 6.11: Coded Simple.

6.2.13 HL7:CO

Coded Ordinal (Figure 6.12) is used to store a coded ordinal such as the code for “2+”. IHC has added the attribute operator to represent things such as “greater” or “less than”, so we can say “greater than 2+”.

\(^4\)The values for these domains need to be defined.
6.14 HL7:IVL Types

Interval Types (Figure 6.13) are used to represent an interval of an ordered datatype. IHC has only defined IVL types for PQ, TS, REAL, and INT, although HL7 allows an interval of any ordered datatype.

6.15 HL7:RTO Types

Ratio Types (Figure 6.14) are used to represent a quantity constructed through the division of a numerator quantity with a denominator quantity. IHC has chosen to only implement a RTO of PQ.

6.16 Misc ASN.1 Declarations

There are a couple of ASN.1 Declarations (Figure 6.15) needed to complete the ASN.1 representation of the HL7 datatypes.
6.2. DESCRIPTION  

CHAPTER 6. HL7 VERSION 3 DATATYPES

IVL−PQ ::= SEQUENCE {
  low [0] PQ OPTIONAL,
  high [1] PQ OPTIONAL }

IVL−TS ::= SEQUENCE {
  low [0] TS OPTIONAL,
  high [1] TS OPTIONAL }

IVL−REAL ::= SEQUENCE {
  low [0] XREAL OPTIONAL,
  high [1] XREAL OPTIONAL }

IVL−INT ::= SEQUENCE {
  low [0] XINT OPTIONAL,
  high [1] XINT OPTIONAL }

Figure 6.13: Interval Types used by IHC.

RTO−PQ ::= SEQUENCE {
  numerator [0] PQ,
  denominator [1] PQ }

Figure 6.14: Ratio Types used by IHC.

Code ::= VisibleString

REAL−Value ::= VisibleString — TODO: constraint

Figure 6.15: Misc. ASN.1 Declarations
Chapter 7

Glossary

7.1 Definitions

cedtype Clinical Element Definition Type. The root element in a CEML Clinical Element definition.

CEM Clinical Element Model. This term covers the individual Clinical Element Definitions as well as the definition of what is a Clinical Element.

CEML Clinical Element Modeling Language. An XML based language used to construct Clinical Element Definitions.

CEO Clinical Element Object. This is a programatic object that is used to manipulate Clinical Element Instance Data. It is analogous to an XML DOM.

Clinical Element A recursive data structure which contains a key code and value choice. The value choice is either other Clinical Elements as children, or an HL7 version 3 datatype. The value choice can then be modified and qualified by other Clinical Elements. This term is used rather losely and can apply to either a Clinical Element Definition or a Clinical Element Instance, so you must know in what context this term is used to fully understand it’s meaning.

Clinical Element Definition The definition of a particular Clinical Element such as a BloodPressurePanel. Instances of this BloodPressurePanel must conform to the rules stated within this definition.

Clinical Element Definition Type See cedtype

Clinical Element Instance An instance of a particular Clinical Element that contains user defined data. For example, a BloodPressurePanel that includes a
systolic blood pressure of 126 mmHg and a diastolic blood pressure of 82 mmHg.

**Clinical Element Modeling Language**  See CEML

**Clinical Element Object**  See CEO

**item**  A child Clinical Element within the value choice of a Clinical Element.

**key**  An attribute within a Clinical Element Instance that is an HL7:CWE. The key’s code links the Clinical Element to a real world coding system.

**mod**  An inner Clinical Element which modifies the content of the value choice in a Clinical Element. The extent of this modification is so great, that the value choice can never be considered independently without simultaneously considering the effect of the modifier on the value choice.

**qual**  An inner Clinical Element which qualifies the content of the value choice in a Clinical Element. The degree to which this qualification changes the meaning of the value choice varies, but it is never to the degree of a modifier. In medical informatics circles, some argue you can never even truly ignore a qualifier, so why make the distinction between a qualifier and a modifier.

**XCE**  A character string that is appended to the end of a Clinical Element type name to create a unique textual code for the vocabulary server. IHC’s vocabulary server called ERTH assigns a unique numeric code (CONCEPT.ID) and a unique textual code (CONCEPT.NAME) to every concept. When we model Clinical Elements and define types, the type names we create are unique within the context of our model, but they are not guaranteed to be unique within the context of the entire vocabulary server. In order to generate unique textual codes for the vocabulary server, we append the suffix XCE to the type name. For example, if we created a Clinical Element type called BloodPressurePanel, then the CONCEPT.NAME in the vocabulary server would assigned BloodPressurePanelXCE.
Chapter 8

To Be Organized

8.1 Versioning

Meetings to be scheduled to finalize plan for Versioning of Clinical Element Models.

8.2 Errors

At one point we had a special Error qualifiers, for ErrorText, and ErrorCoded. We were going to create a special nullFlavor for the HL7 datatypes, which would direct the user to look in the appropriate Error location. But this was deemed unnecessary and Errors are now handled like this.

   Errors

   1. Any text targeted for a coded value will go into originalText.

   2. Null values will just be handled with nullflavor.

   3. to be continued . . .

8.3 Linking screen fields to data fields

This functionality will most likely take place outside of the Clinical Element models in some sort of mapping file like the Profiler uses today. Need to schedule meetings with Scott and Adam to discuss requirements.
8.4 TypeOid

The TypeOid is a Clinical Element type path, that exists one to one for any Clinical Element type. It is not required in an instance, but can be requested to be filled in when Clinical Elements are requested from the data store. The TypeOid makes it easier to treat siblings of a common type as that common type.

8.5 Tidbits

1. Don’t pre-coordinate Subject into Type or Key; always design Family History related information in a general way using Subject.

2. Don’t pre-coordinate Negation into Type and Key.

8.6 Interfaces

Future meetings to discuss the role of interfaces similar to Java interfaces.

8.7 ReportFlag

The purpose of ReportFlag is to flag individual pieces of data as to whether they will show up on generated reports. The main use case is Microbiology data, where susceptibility on experimental drugs is performed by the laboratory, but this information is not to be displayed to the physician.

8.8 Labels

The purpose of labels is to allow an alternate user defined classification to Clinical Elements. Labels were previously called Classification. We are still struggling how to allow multiple labels and still be able to query based on the label. One label would be easy, but if we allow multiple labels, the required join in an underlying relational database will probably not perform.